SONDERHEFT 65 Markt&Technik ÖS 120,-/Sfr. 16,-/Lit. 16000 hfl. 21,-/dkr. 75,-/fmk 62,- DM 16,-

DISKETTE IM HEFT

64'er

TIPS & TOOLS

Drucker-Basic

58 neue Befehle zur Printer-Steuerung

Grafik-Wandler

Konvertiert jedes Multicolorbild: Logo-Painter

Speicher-Safari

Entdecken Sie die Zeropage!



ÜBER 60 PROGRAMME AUF DISKETTE

64'er



Das C 64 Tips & Tricks Poster

Umschlagseiten von den Heftklammern lösen!

Die zehn besten »Einzeiler«

Old: Gibt man folgende Zeile im Direktmodus (ohne Zeilennummer) ein, so wird ein durch NEW oder Reset gelöschtes Basic-Programm wieder hergestellt.

```
0 POKE 2050,8:SYS 42291:POKE 46,PEEK(35)-(PEEK(781)
>253):POKE 45,PEEK(781)+2 AND 255:CLR    <167>
```

Merge: Hat man ein Programm im Speicher und möchte ein zweites an dieses anhängen, so gibt man einfach folgende Zeile im Direktmodus (ohne Zeilennummer) ein.

```
0 A=PEEK(45)+256*PEEK(46)-2:POKE 44,
A/256:POKE 43,A-PEEK(44)*256    <012>
```

Anschließend lädt man das zweite Programm mit LOAD und gibt dann POKE 43,1:POKE 44,8 ein.

Directory: Will man sich das Directory einer Diskette ansehen ohne ein im Speicher liegendes Programm zu zerstören, so gibt man einfach folgende Zeile ein.

```
0 GET#1,A$: A=ASC(A$+"{HOME}"):PRINT CHR$
((A=130 AND 13 OR((31<A AND A<95)AND A)));
:GOTO 0    <007>
```

Aufgerufen wird diese Zeile mit OPEN 1,8,2,"$":GOTO 0.

Renumber: An ein Programm angehängt bewirkt folgender Einzeiler, daß die Zeilen des Programms neu durchnumeriert werden.

```
0 FOR A=2049 TO PEEK(45)+PEEK(46)*256-3:POKE A+2,Z:
POKE A+3,0:A=PEEK(A)+PEEK(A+1)*256-1:Z=Z+1:
NEXT    <092>
```

Input mit Komma: Dieser Einzeiler ersetzt den INPUT-Befehl, hat jedoch den Vorteil, daß auch Satzzeichen wie Komma, Doppelpunkt und Strichpunkt eingegeben werden können.

```
0 SYS 42336:XX$="":FOR II=512 TO 600:AA=PEEK(II):
IF AA THEN XX$=XX$+CHR$(AA):NEXT    <237>
```

Print at: Mit Hilfe dieser kleinen Routine kann man den Cursor gezielt auf bestimmte Bildschirmpositionen plazieren. Vor dem Aufruf sollten die Variablen Z, S und T$ die Werte für Zeile-1, Spalte beziehungsweise den zu druckenden String enthalten.

```
0 POKE 214,Z:PRINT:POKE 211,S:PRINT T$    <141>
```

Floppy on/off: Dieser Einzeiler prüft, ob das angeschlossene Floppylaufwerk eingeschaltet ist und gibt eine entsprechende Meldung aus.

```
0 POKE 768,61:OPEN 1,8,15:CLOSE 1:POKE 768,139:
IF ST=-128 THEN PRINT"FLOPPY OFF!"    <221>
```

Drucker on/off: Folgende Programmzeile überprüft, ob der angeschlossene Drucker eingeschaltet ist. Wenn nicht, wird eine Meldung ausgegeben.

```
0 POKE 768,61:OPEN 1,4:PRINT#1:CLOSE 1:POKE 768,139:
IF ST=-128 THEN PRINT"PRINTER OFF!"    <031>
```

Floppyspeeder: Diese kleine Routine verkürzt die Zugriffszeiten der Floppy 1541. Alle Kopfbewegungen werden um den Faktor vier beschleunigt!

```
0 OPEN 1,8,15,"M-W"+CHR$(7)+CHR$(28)+CHR$(1)+CHR$
(15):CLOSE 1    <098>
```

Kopierschutz: Dieser Einzeiler erzeugt auf dem gewünschten Track einer Diskette (Variable T) einen READ ERROR 21. Mit normalen Back-up-Kopierprogrammen kann eine mit diesem Einzeiler behandelte Diskette dann nicht mehr kopiert werden.

```
0 OPEN 1,8,15:OPEN 2,8,2,"#":PRINT#1,"U1 20";
T;0:PRINT#1,"M-E"CHR$(163)CHR$(253):CLOSE 1:
CLOSE 2    <016>
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	BRK	ORA ,X				ORA ZP	ASL ZP		PHP	ORA #	ASL A			ORA	ASL	
1	BPL	ORA ,Y				ORA ZP, X	ASL ZP, X		CLC	ORA ,Y				ORA ,X	ASL ,X	
2	JSR	AND ,X			BIT ZP	AND ZP	ROL ZP		PLP	AND #	ROL A		BIT	AND	ROL	
3	BMI	AND ,Y				AND ZP, X	ROL ZP, X		SEC	AND ,Y				AND ,X	ROL ,X	
4	RTI	EOR ,X				EOR ZP	LSR ZP		PHA	EOR #	LSR A		JMP	EOR	LSR	
5	BVC	EOR ,Y				EOR ZP, X	LSR ZP, X		CLI	EOR ,Y				EOR ,X	LSR ,X	
6	RTS	ADC ,X				ADC ZP	ROR ZP		PLA	ADC #	ROR A		JMP I	ADC	ROR	
7	BVS	ADC ,Y				ADC ZP, X	ROR ZP, X		SEI	ADC ,Y				ADC ,X	ROR ,X	
8		STA ,X			STY ZP	STA ZP	STX ZP		DEY		TXA		STY	STA	STX	
9	BCC	STA ,Y			STY ZP, X	STA ZP, X	STX ZP, Y		TYA	STA ,Y	TXS			STA ,X		
A	LDY #	LDA ,X	LDX #		LDY ZP	LDA ZP	LDX ZP		TAY	LDA #	TAX		LDY	LDA	LDX	
B	BCS	LDA ,Y			LDY ZP, X	LDA ZP, X	LDX ZP, Y		CLV	LDA ,Y	TSX		LDY ,X	LDA ,X	LDX ,Y	
C	CPY #	CMP ,X			CPY ZP	CMP ZP	DEC ZP		INY	CMP #	DEX		CPY	CMP	DEC	
D	BNE	CMP ,Y				CMD ZP, X	DEC ZP, X		CLD	CMP ,Y				CMP ,X	DEC ,X	
E	CPX #	SBC ,X			CPX ZP	SBC ZP	INC ZP		INX	SBC #	NOP		CPX	SBC	INC	
F	BEQ	SBC ,Y				SBC ZP, X	INC ZP, X		SED	SBC Y				SBC ,X	INC ,X	

▲ **Der Befehlssatz der C-64-Mikroprozessoren 6502/6510 auf einen Blick. Die Zahlenreihe seitlich links gibt das Hi-Nibble, die obere Leiste das Low-Nibble des Codes nach dem Assemblieren an.**

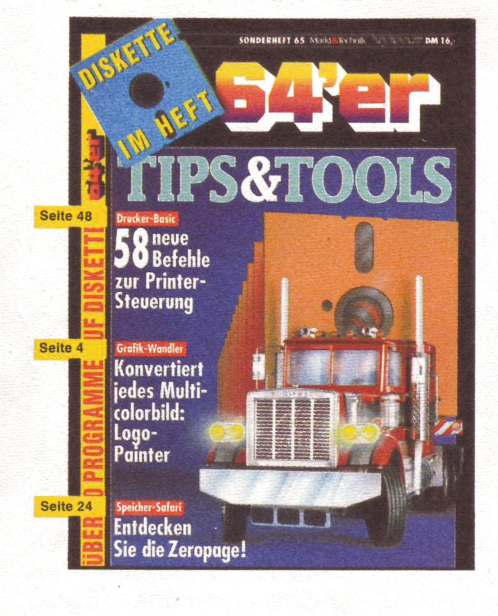
Schreib-Register

Adresse	Reg.	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2
54272	0	F7	F6	F5	F4	F3	F2
54273	1	F15	F14	F13	F12	F11	F10
54274	2	P7	P6	P5	P4	P3	P2
54275	3	unbenutzt	unbenutzt	unbenutzt	unbenutzt	P11	P10
54276	4	Rauschen	Rechteck	Sägezahn	Dreieck	Test	Ringmodulat.
54277	5	Attack 3	Attack 2	Attack 1	Attack 0	Decay 3	Decay 2
54278	6	Sustain 3	Sustain 2	Sustain 1	Sustain 0	Release 3	Release 2
54279	7	F7	F6	F5	F4	F3	F2
54280	8	F15	F14	F13	F12	F11	F10
54281	9	P7	P6	P5	P4	P3	P2
54282	10	unbenutzt	unbenutzt	unbenutzt	unbenutzt	P11	P10
54283	11	Rauschen	Rechteck	Sägezahn	Dreieck	Test	Ringmodulat.
54284	12	Attack 3	Attack 2	Attack 1	Attack 0	Decay 3	Decay 2
54285	13	Sustain 3	Sustain 2	Sustain 1	Sustain 0	Release 3	Release 2
54286	14	F7	F6	F5	F4	F3	F2
54287	15	F15	F14	F13	F12	F11	F10
54288	16	P7	P6	P5	P4	P3	P2
54289	17	unbenutzt	unbenutzt	unbenutzt	unbenutzt	P11	P10
54290	18	Rauschen	Rechteck	Sägezahn	Dreieck	Test	Ringmodulat.
54291	19	Attack 3	Attack 2	Attack 1	Attack 0	Decay 3	Decay 2
54292	20	Sustain 3	Sustain 2	Sustain 1	Sustain 0	Release 3	Release 2
54293	21	unbenutzt	unbenutzt	unbenutzt	unbenutzt	unbenutzt	GF 2
54294	22	GF 10	GF 9	GF 8	GF 7	GF 6	GF 5
54295	23	Resonanz 3	Resonanz 2	Resonanz 1	Resonanz 0	Filtex	Filter 3
54296	24	Aus	Hochpass	Bandpass	Tiefpass	L 3	L 2

Lese-Register

Adresse	Reg.	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2
54297	25	Pot X 7	Pot X 6	Pot X 5	Pot X 4	Pot X 3	Pot X 2
54298	26	Pot Y 7	Pot Y 6	Pot Y 5	Pot Y 4	Pot Y 3	Pot Y 2
54299	27	07	06	05	04	03	02
54300	28	H7	H6	H5	H4	H3	H2

INHALT



Tips & Tricks

Grafik

Floppy

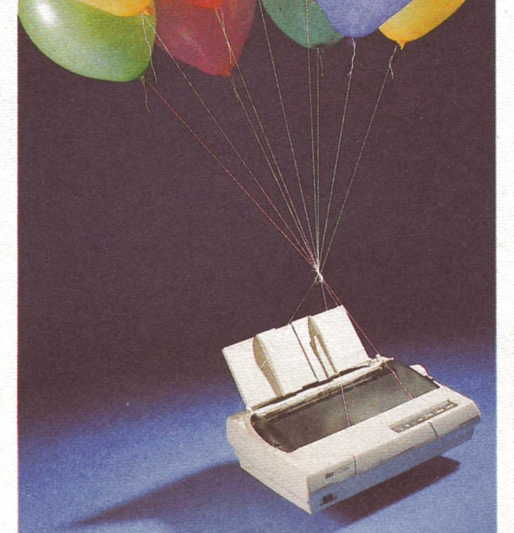
Programmieren

Drucker

Sonstiges

Alle Programme aus Artikeln mit einem ■-Symbol finden Sie auf der beiliegenden Diskette (Seite 19).

MDG Convert:
Ein superschneller Cross-Converter für alle Multicolor-Files
Seite 8



Drucker-Basic:
Noch nie war Drucken so leicht
Seite 48

Logo-Painter Multicolorgrafik als Zeichensatz

Kein Pinselstric

Das Programm ist fast fertig, nur noch ein gutes Logo für das Titelbild fehlt. Dummerweise ist nicht mehr genügend Speicher frei, 41 Blocks für die Farbgrafik sind nicht mehr drin. Der Logo-Konverter löst das Problem, indem er hochauflösende Farbgrafiken trickreich in Zeichensatz konvertiert. Ein kleines Malprogramm ist auch eingebaut.

von Patryk Logiewa

Wie wäre es mit einem Logo (s. Textkasten) für Ihr Programm? Diese Einstellungsbilder bewundert man in den Crakker-Demos. Problematisch ist nur, daß keine 41 Blöcke für eine Multicolor-Grafik ins Programm passen. Nachladen? Wäre eine Möglichkeit, kostet aber viel Zeit, und auch Platz auf der Diskette und der ist ohnehin knapp. Als Lösung bietet sich eine Umcodierung des Bildes an, eine »Zeichensatzgrafik«. In der Tat wird dieser Weg oft dann gegangen, wenn der Speicher knapp oder großer Wert auf Geschwindigkeit gelegt wird.

Im Unterschied zu anderen Programmen (z.B. Hitoloko aus Sonderheft 55) ermöglicht der Logo-Painter spielend einfach die Übernahme und Wandlung von Multicolor-Bildern. Darüber hinaus ist noch ein kleines Malprogramm eingebaut, mit dem Korrekturen an fertigen Bildern vorgenommen werden können. Zur Kreation der Grafiken eignen sich die großen Malprogramme, wie OCP-Art-Studio, Koala-Painter, Blazing-Paddles oder natürlich Amica-Paint. Anschließend wird mit Logo-Painter konvertiert und dabei werden zwei Files auf Diskette erzeugt: ein Zeichensatz und der dazugehörige Bildschirminhalt. Eine große Hilfe für die Speicherfunktionen ist die eingebaute RAM-Disk, in der Files abgelegt werden können. Sie brauchen dazu keine Zusatzhardware, vielmehr wurde ein Bereich des C64-Speichers extra für diesen Zweck reserviert. Hier lagern Sie Zwischenergebnisse bei der Arbeit an Ihrem Logo.

Der Painter ist aus Geschwindigkeits- und Komfortgründen vollständig in Maschinensprache geschrieben und kann wie ein Basic-Programm geladen, gestartet und auch kopiert werden. Zwei Versionen befinden sich auf der beiliegenden Diskette. Die lange Version enthält ein Beispiellogo und eine englische Kurzanleitung. Bei der Short-Version ist der Bildschirm nach dem Start gelöscht und es wurde auf die eingebaute Anleitung verzichtet.

Die lange Version laden Sie mit:

```
LOAD "LP3+(FULL),8"
```

die Kurzversion mit

```
LOAD "LP3+(SHORT)",8
```

Beide werden nach dem Start mit RUN sekundenschnell entpackt. Danach erscheint das Titelbild mit dem Hauptmenü (Abb. 1).

Hier stehen Ihnen auf Tastendruck folgende Funktionen zur Verfügung:

<I> Instructions

...gibt auf mehreren Bildschirmseiten kurze Informationen zur Bedienung und Funktionsweise des Logo-Painters (in Englisch). Bei der Kurzversion fehlt diese Option.

<E> Editor

...eignet sich hauptsächlich zur Nachbearbeitung eines konvertierten Bildes. Mit <E> wird er aus dem Hauptmenü gestartet. Die Steuerung des Cursors (Pfeilform) erfolgt wahlweise über Joystick an Port 2 oder mit einer Maus. Bei Betätigung des Feuerknopfes werden Punkte in der aktuellen Cursor-Farbe gesetzt.

Zwei Grundfunktionen machen den Editor zu einem universellen Werkzeug. Der Grafikeditor zeigt das konvertierte oder gezeichnete Bild und der Lores-Editor läßt eine Bearbeitung mit dem originalen Zeichensatz zu (Layout). Umgeschaltet wird mit <S>.

Wählen Sie unter folgenden Tastenfunktionen:

F1 – Farbe Nr. 1 (Farb-RAM) wird aktuelle Cursor-Farbe.

F3 – Farbe Nr. 2 ($D022)

F5 – Farbe Nr. 3 ($D023)

DEL – Hintergrundfarbe ($D021) damit werden gesetzte Punkte gelöscht

F2 – erhöht Farbe Nr. 1 um eins

F4 – erhöht Farbe Nr. 2 um eins

F6 – erhöht Farbe Nr. 3 um eins

F7 – erhöht Farbe Nr. 4 um eins

SPACE – ein Zeichen kopieren. Bewegen Sie zuerst den Cursor auf das zu kopierende Zeichen und betätigen Sie die Leertaste. Danach bringen Sie den Cursor auf die neue Position und drücken Feuer.

CRSR UP/DOWN – scrollt vertikal durch den Zeichensatz. Logos können sich über bis zu zwei Bildschirmseiten erstrecken.

CRSR LEFT/RIGHT – die Zeile unter dem Cursor wird nach links oder rechts gescrollt. So läßt sich Text zentrieren oder randbündig machen.

* – hiermit wird das letzte, nicht verwendete Zeichen (Nummer 255) invertiert. Nützlich ist dies vor allem zur Abschätzung des Umfangs eines Logos.

CLR – löscht die gesamte Zeichensatzdefinition, während das Layout des Bildschirms unverändert bleibt. Um auch diesen Bereich zu löschen, wird mit <S> der Lores-Editor aktiviert und dort mit <CLR> gelöscht.

h zuviel

HOME – stellt das Fenster auf den Anfang des Editorbereichs (Beispiel $1800) zurück.

S – schaltet zwischen Screen- und Zeichensatzeditor um.

I – Umschaltung des Eingabegerätes: Joystick oder Maus. Das aktive Gerät wird rechts am unteren Bildschirmrand symbolisch dargestellt.

RUN/STOP – Rückkehr zum Hauptmenü.

Im untersten Bildschirmbereich sind wichtige Betriebsparameter eingeblendet (Abb. 2). Von links nach rechts sehen Sie:

1. den Zoom-Bereich, in dem das aktuelle Zeichen doppelt vergrößert zu sehen ist.

2. Die Anzahl der benutzten Zeichen.

3. Eine Anzeige des Speicherumfangs in hexadezi-



maler Darstellung, zur Abschätzung der Start/Endadressen des Bildschirms.

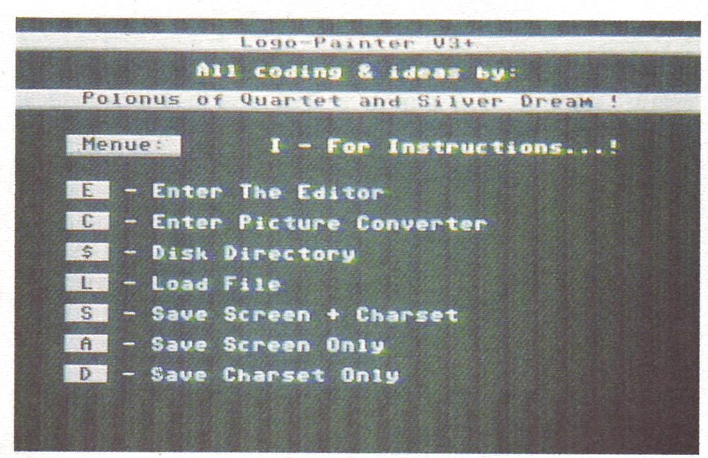
4. Die Anzeige der Nummer des Zeichens unter dem Cursor.

5. Das Piktogramm für das gerade aktive Eingabegerät oder der Kopiermodusindikator. Wenn eine Analogmaus angeschlossen ist, läßt sich mit dem rechten Maustaster die Anzahl der Freiheitsgrade einstellen. Dieser Wert ist ebenfalls im Piktogramm ersichtlich.

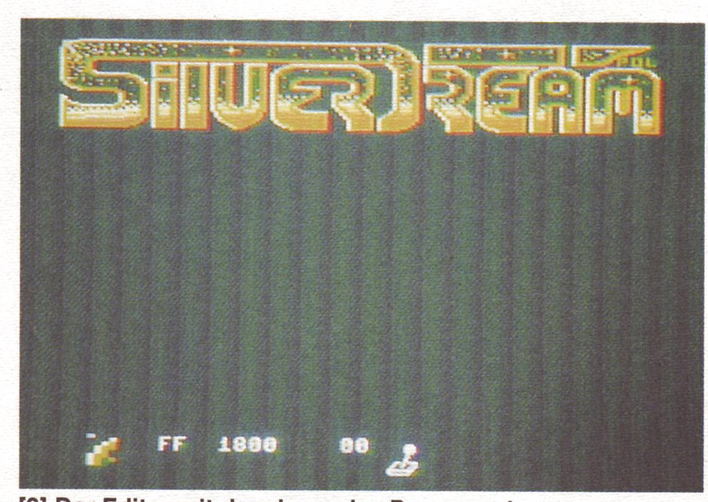
Die beschriebenen Funktionen sind sowohl im Grafikeditor wie auch – mit Einschränkungen – im Lores-Editor gültig. Hier setzen Sie durch Druck auf Feuer Zeichen, die mit <DEL> auch wieder gelöscht werden können. Mit Hilfe dieses Programmteils lassen sich sehr bequem auch größere Teile des Logos kopieren oder verschieben.

<C> Converter

Ein Tastendruck auf <C> ruft den Grafikwandler auf. Er kopiert fertige Multicolor-Bilder ins Logo-Painter-Format um. Verwendbar sind Files in den Formaten Koala-Painter, Advanced Art Studio und Blazing Paddles. Da sich im Editor nur Funktionen zur Überarbeitung von Bildern befinden, bietet es sich an, Logos mit den komfortablen Funktionen eines Malprogramms zu zeichnen. Auch andere Programme, wie unser Amica-Paint (Sonderheft 45), lassen sich dazu verwenden. Konvertieren Sie diese Files mit dem MDG-Konvert (S. 8) in ein für den Logo-Painter lesbares Format.



[1] Das Hauptmenü des Logo-Painter (LONG)



[2] Der Editor mit dem Logo des Programmierers

Nachdem Sie vom Hauptmenü aus den Konverter mit <C> gestartet haben, werden Sie nach einem Filenamen gefragt. Geben Sie den Namen ein, unter dem Ihr Werk auf Diskette gespeichert ist und bestätigen mit <RETURN>. Die Suffixe (Kennung) der Malprogramme ».MPIC«, »PI.« usw. sind wegzulassen. Sie werden von »Logo-Painter« automatisch angefügt. Dazu müssen Sie im nächsten Schritt das Format der Grafik durch eine der Zifferntasten <1>, <2> oder <3> bestätigen. Erst danach erfolgt das Laden. Wird das angegebene File auf der eingelegten Diskette nicht gefunden, erscheint eine entsprechende Meldung. Wundern Sie sich nicht, wenn im Programm auf verschiedene Arten der Fehlbedienung etwas lässig reagiert wird, z.B. bei falschen Filenamen als »lamer« (Anfänger).

Ist alles korrekt eingegeben, lädt der Konverter die Grafik und zeigt eine Statistik der verwendeten Farbtöne in Pixel an (Abb. 3). Anhand dieser Aufstellung läßt sich die Komplexität abschätzen. Sie dürfen in der Grafik maximal drei verschiedene Farben verwenden. Sind es mehr, verwarnt Sie das Programm in seiner gewohnt saloppen Art (»don't you lamer know the maximum of three colours! Try to convert it anyway? (y/n)«). Das Bild kann trotzdem mit Einschränkungen gewandelt werden. Entweder Sie bestätigen mit <y>, dann werden die drei häufigsten Farbtöne übernommen, oder Sie lassen das Bild mit <n>, ohne Rücksicht auf Verluste so wie es im Speicher steht, übertragen.

Eine Einschränkung ist noch in Kauf zu nehmen: Sie betrifft die Werte der Farben. Zwei können jeden beliebigen Wert annehmen, die dritte muß im Bereich von 0 (schwarz) bis 7 (gelb) liegen. Halten Sie sich nicht daran, kann eine der drei Farben in den benötigten Bereich umgestellt werden.

Jetzt erst beginnt das eigentliche Konvertieren. Ist das Bild zu komplex, und genügen die 256 zur Verfügung stehenden Zeichen nicht, erscheint die Meldung:

`NOT ENOUGH CHARACTERS...!!!`

und der Konverter bricht seine Arbeit ab. Solche Bilder lassen sich mit dem »Logo-Painter« leider nicht komplett konvertieren. Der bisher erzeugte Teil läßt sich natürlich weiterverwenden. Achten Sie deshalb bereits beim Malen des Logos auf möglichst geringe Details und große ausgefüllte Flächen.

Was ist ein »Logo«

Da das hier vorgestellte Programm schon »Logo-Painter« heißt, und auch sonst der Begriff »Logo« recht oft in diesem Artikel auftaucht, hier eine Definition für diese Bezeichnung.

Ursprünglich stammt der Begriff aus der Sprache der Werbefachleute. Ein Unternehmen hat ein Logo, ein Symbol mit Wiedererkennungswert. Die Deutsche Bank AG etwa hat ein Quadrat mit dem Schrägbalken als Logo.

In der EDV wurde »Logo« von Crackern geprägt. Sie zeigen sich immer unter einem bestimmten Schriftzug. Vor einem Programm, das von diesen Leuten gecrackt (entschützt) wurde, findet sich im Vorspann dann unverkennbar das dazugehörige »Logo«. Konkret handelt es sich dabei meistens um ein Grafikbild in etwa der Breite des Bildschirms, von der Höhe her etwa ein Drittel des Bildschirms. Meistens sind diese Logos dann noch kunstvoll grafisch verziert oder animiert und bewegt.

Das Titelbild beispielsweise zeigt das Logo des Autors von »Logo-Painter« Patryk Logiewa, der unter dem Künstlernamen »Silver Dream« agiert.

(Nikolaus Heusler/gr)

Kurzinfo: Logo-Painter

Programmart: Grafik-Hilfsprogramm
Laden: LOAD "LP3+(LONG)",8,0
Start: nach dem Laden RUN eingeben
Steuerung: Joystick, Maus und/oder Tastatur
Besonderheiten: Konverter zur Wandlung von hochauflösenden Multicolor-Bildern in Zeichensatzgrafiken
Länge in Blocks: lang: 45, kurz: 29
Programmautor: Patryk Logiewa

Nach erfolgreicher Konvertierung erscheint auf dem Bildschirm noch die Anzahl der benötigten Zeichen. Danach führt ein beliebiger Tastendruck ins Hauptmenü zurück.

<$> Directory

Mit <$> erhalten Sie die Anzeige des Inhaltsverzeichnisses der eingelegten Diskette. Nach der Ausgabe führt ein beliebiger Tastendruck wieder ins Hauptmenü.

<L> Load File

Eine Datei läßt sich wahlweise aus der eingebauten RAM-Floppy oder wie üblich von Diskette laden. Beachten Sie bei Diskettenoperationen, daß der Computer ans Ende noch die Kennung »/V3+« hängt. Soll ein File ohne diese Kennung geladen werden, beenden Sie den Filenamen mit einem Sternchen. Logo-Painter-Dateien werden automatisch an die richtige Speicherposition geladen. Anhand der Ladeadresse unterscheidet man zwischen Grafiken und Zeichensätzen. Die RAM-Floppy benötigt keine Namensangabe.

<S> Save Screen and Charset

...dient zum Speichern von Zeichensatz und dazugehörigem Lores-Bildschirm auf RAM-Floppy oder Diskette. Auch hier kommt die Kennung an das Ende des Dateinamens. Das Ergebnis ist ein einziges File zu 17 Blöcke mit der Ladeadresse 6144. Es enthält ab Adresse 6144 die Bild- und ab 8192 die Zeichensatzdaten. Gegenüber zwei getrennten Dateien (s.u.) sparen Sie einen Block auf Diskette. Als Besonderheit lassen sich diese Komplett-Files von Basic aus mittels SYS direkt einschalten. Wie das funktioniert, lesen Sie im Textkasten »In der Praxis«.

<A> Save Screen only

Soll nur der Lores-Schirm gespeichert werden, rufen Sie diese Option auf. Das erzeugte File enthält den Bildschirm, die Ladeadresse ist 6144. Die Bedienung erfolgt wie oben beschrieben. Es wird ein einzelnes File erzeugt, das die Bild-Daten enthält.

<D> Save Charset only

In dieser Option wird nur der Zeichensatz Ihrer Grafik gespeichert. Die Bedienung erfolgt wie in »Save Screen & Charset« beschrieben. Es wird ein einzelnes File mit neun Blöcken und der Startadresse 8192 im Standardformat erzeugt.

```
You have used the following colours:

00 - Black ........        Not used...
01 - White ........4144    multi-pixels
02 - Red ..........        Not used...
03 - Cyan .........        Not used...
04 - Purple .......        Not used...
05 - Green ........        Not used...
06 - Blue .........25360   multi-pixels
07 - Yellow .......        Not used...
08 - Orange .......        Not used...
09 - Brown ........        Not used...
10 - Light Red ....        Not used...
11 - Dark Grey ....        Not used...
12 - Medium Grey ..        Not used...
13 - Light Green ..        Not used...
14 - Light Blue ...        Not used...
15 - Light Grey ...        Not used...

Together There is..29504 multi-pixels
      set in the whole picture...
```

[3] Die Farbstatistik zeigt die Farbverteilung der Grafik

Das Programm kann durch <RUN/STOP-RESTORE> oder <RESET> beendet werden. Ein Neustart erfolgt mit

```
SYS 16384
```

Sie sind dann wieder im Hauptmenü.

Demo

Damit Sie sich einen groben Eindruck verschaffen können, was mit der Pseudo-Hires-Programmierung für tolle Effekte möglich werden, befindet sich ein Demoprogramm mit auf Diskette. Dieses wird mit

```
LOAD "4.*",8
```

geladen und mit RUN gestartet. Nach einigen Sekunden Entpackzeit zeigt der Programmierer hier, was er wirklich kann. Die einzelnen Teile dieses »Demos« lassen sich mit der Leertaste weiterblättern. Nach dem letzten Teil steigt das Programm aus. Es wird versucht, ein Programm mit dem Filenamen »5.*« von Diskette nachzuladen und mit SYS zu starten. Da das hier natürlich nicht klappt, steigt der Rechner aus.

(Nikolaus M. Heusler/gr)

In der Praxis

Damit Sie die erzeugten Files in eigene Programme einbauen können, erleben Sie anhand eines konkreten Beispiels den Weg eines Logos. Alles beginnt damit, daß Sie in einem Malprogramm Ihrer Wahl das Logo zeichnen. Verwenden Sie nicht mehr als drei Farben (zuzüglich Hintergrundfarbe), eine davon nur im Bereich von 0 bis 7, und gestalten Sie das Bild nicht allzu detailliert. Konvertieren Sie es danach mit MDG-Konvert (S. 8) in eins der Formate: Blazing Paddles, Koala-Painter oder Advanced OCP Studio. Danach speichern Sie es auf Diskette. Dort liegt es jetzt als 41 Block lange Multicolor-Datei vor. Laden und starten Sie wie oben beschrieben den Logo-Painter und konvertieren Sie das Bild.

Nachdem Ihr Werk (hoffentlich erfolgreich) gewandelt wurde, können Sie es bei Bedarf im Editor nachbearbeiten. Sind Sie mit dem Ergebnis zufrieden, speichern Sie es mit den Funktionen <A> und <D> in zwei Dateien zu je neun Blöcke ab. Geben Sie bei <A> als Filenamen SCREEN ein, bei <D> CHAR. Der Platzgewinn auf Diskette ist beträchtlich: 41 Blöcke sind auf 18 geschrumpft - ohne daß auch nur ein Pixel des Logos verlorengegangen ist! 56 Prozent Speicherplatz sind gewonnen.

Die erzeugten Files lassen sich nun nachladen. Das Bild liegt danach etwas ungünstig ab 6144 im Speicher (der VIC kann diesen Bereich nicht direkt einzublenden). Darum empfiehlt es sich, die Bilddaten nach 1024 in den Bildschirmspeicher umzukopieren. Eine kurze FOR..NEXT-Schleife genügt da schon:

```
FORI=0TO999:POKE1024+I,PEEK(6144+I):NEXT
```

Ein ausgefuchster Programmierer lädt die Datei einfach ohne Berücksichtigung der vorgegebenen Ladeadresse relativ nach 1024 (vgl. weiter unten abgedruckte Maschinenspracheroutine). Der Zeichensatz beginnt bei 8192, dort kann er mit

```
POKE 53272,24
```

sichtbar gemacht werden. Das Einschalten des Multicolor-Betriebes erfolgt mit

```
POKE 53270,216
```

Abschließend müssen noch die Farben korrekt gesetzt werden. Sie finden diese Schritte zusammengefaßt im Demoprogramm »EINBINDUNGS-BSP.«. Es lädt zwei Dateien auf der Diskette im Heft automatisch nach und zeigt sie an.

In einer eigenen Routine ändern Sie einfach in Zeile 30 den Filenamen der Bild- und in 40 den Namen der Zeichensatz-Datei. Wie Sie sehen, beenden wir hier den Filenamen mit einem Stern, somit kann die Kennung »/V3+« des »Logo-Painters« ignoriert werden. Solche kleinen Tricks helfen, Speicherplatz einzusparen.

Es gibt noch eine zweite, äußerst interessante Methode, fertige Logos einzublenden. Dazu laden Sie ein mit der S-Funktion gespeichertes Kompaktfile absolut und »starten« es sodann mit

```
SYS 10240
```

Sofort erscheint das erwünschte Logo auf dem Bildschirm, dabei werden sogar die korrekten Farben gesetzt. Danach wird ggf. der nächste Basic-Befehl ausgeführt.

```
SYS 65409
```

beendet den Spuk wieder. Wie das in der Praxis anzuwenden ist, zeigt folgendes kleines Basic-Programm:

```
10 IF A=0 THEN A=1: LOAD "NAME",8,1
20 SYS 10240
30 REM weiter im Programm
```

Da der LOAD-Befehl ein Programm ständig neu starten würde, verhindern wir in Zeile 10 mit der Variablen A eine Endlosschleife.

von Oliver Stiller

Szenenwechsel

MDG wandelt alle Formate

Jedes Malprogramm hat spezifische Vorzüge. Doch was tun, wenn ein Werk mit Koalapainter gezeichnet und mit Amica Paint weiterbearbeitet werden soll? – Ganz einfach, Sie wandeln es mit MDG-Konvert.

Files unterschiedlicher Malprogramme lassen sich normalerweise nicht untereinander austauschen. Es funktioniert nur mit einem einfach zu bedienenden Hilfsprogramm. MDG-Konvert lädt die Formate Koalapainter, Amica Paint, Blazing Paddles, Advanced Art Studio, Paint Magic, Bitmap, Zeichensatz plus Video-RAM (mit Farb-RAM) und Sprites und speichert sie in einem anderen der genannten Formate. Geladen und gestartet wird der Konverter von der beiliegenden Diskette durch:

```
LOAD"MDG-KONVERT",8
```

und RUN. Das folgende Titellogo läßt sich mit einem beliebigen Tastendruck abbrechen. Anschließend sehen Sie das Hauptmenü. Durch Tastendruck auf die dort angegebene Taste rufen Sie Hauptfunktionen auf:

F1 – Input

Lädt Grafiken von Diskette. Die Files werden dabei unabhängig von der gespeicherten Startadresse richtig geladen.

A – Clear Screen – löscht den Bildschirm und führt zurück ins Hauptmenü.

K – Directory – erzeugt eine Inhaltsanzeige der eingelegten Diskette. Nach je einer Bildschirmseite stoppt die Ausgabe. Weiter geht's mit beliebigem Tastendruck.

L – DOS Command – sendet Befehle zur Diskettenstation.

G – simple Bitmap – lädt 8000 Byte Grafikdaten, die als Bitmuster im Format des VIC-Chips vorliegen müssen. Sie werden zuerst nach vier Farben gefragt, die den einzelnen Bitkombinationen der Grafik entsprechen. Anschließend erwartet das Programm den Dateinamen. Das File sollte nicht länger als 33 Blöcke (8192 Byte) sein.

I – Video-RAM + Charset – auch hier ist zunächst die Eingabe der vier zu verwendenden Farben nötig. Als »Video-RAM« wird eine Datei mit vier bis fünf Blöcken (1000 bis 1024 Byte) erwartet. Der Zeichensatz (Charset) kann bis zu neun Blöcke (256 Zeichen) lang sein.

H – Video-RAM + Charset + Color-RAM – entspricht weitgehend der Option »I«, es wird jedoch zusätzlich der Farbspeicher geladen.

J – Sprites – die geladenen Sprites werden auf der gerade sichtbaren Grafik plaziert. Entspricht die aktuelle Hintergrundfarbe nicht einer der eingegebenen Farben, wird der Hintergrund neu eingefärbt.

Das Plazieren der Sprites auf dem Bildschirm kann automatisch <A> oder individuell <I> geschehen. Bei Automatik werden bis zu 80 Sprites in einem Raster angeordnet. Falls sich mehr als 80 Sprites in einer Datei befinden, werden die überzähligen ignoriert.

Bei Individuell sind folgende Tasten belegt:

<I>,<J>,<K>,<M> – bewegen den Cursor in Ein-Pixel-Schritten.

<Cursor-Tasten> – der Cursor wird in 12 x 21-Pixel-Schritten in die entsprechende Richtung bewegt.

C,E,B,D und F – laden das angegebene Grafikformat. Die Formatkennung wird angezeigt und automatisch ergänzt.

F3 – Modify

Läßt ein Verändern der Farben <A>, bzw. eine Farbstatistik <B> zu.

F5 – Output

Enthält die Speicherroutinen für die konvertierten Grafiken.

K und L – diese Funktionen entsprechen denen unter »F1 - Input«.

C,E,B und F – Speichert nach Eingabe des Filenamens in dem gewählten Format. Die Kennung des Formats wird angezeigt und automatisch hinzugefügt.

A – Epson Hardcopy – gibt eine Hardcopy auf Epson-kompatiblen Druckern aus. Dabei werden Graustufen simuliert. Bevor die Hardcopy startet, müssen Sie die Frage nach einem Linefeed (Zeilenvorschub) mit »Y« für Ja oder »N« für Nein beantworten (bitte ausprobieren).

D – save Paint Magic – Da Paint Magic nicht alle Farbmöglichkeiten ausnützt, läßt sich nicht alles in diesem Format speichern. Sie erhalten dann eine Fehlermeldung. Die Bildschirmposition beim Abbruch wird angezeigt.

G – save simple Bitmap – Geben Sie nacheinander die Farben für die Bitkombinationen 00,01 und 10 ein. Die vierte Farbe wird automatisch zugeordnet. Bei mehr als vier Farben erscheint eine Fehlermeldung.

H – save video-RAM + charset – wandelt Bitmapgrafik in Zeichensatzgrafik. Die Grafik darf nur vier Farben enthalten. Zur Eingabe verfahren Sie wie bei »G«. Character »0« wird in jedem Fall als Leerzeichen definiert. Nach der Anwahl durch <H> haben Sie die Auswahl zwischen <B> (Normalmodus) und <A> (Playfield-Modus). Der Normalmodus wandelt ein Bild, der Playfieldmodus mehrere unter einen einzigen Zeichensatz. Beim nächsten Aufruf können Sie unter »next« das nächste oder unter »last« das Bild umwandeln. Danach wird der Zeichensatz gespeichert. Ein Zeichensatz hat die Startadresse $8000, das Video-RAM $C000.

I – save video + char + color-RAM – entspricht weitgehend »H«. Das Bild darf dabei mehr als vier Farben enthalten. Im Farb-RAM darf nur die Farbe 0-7 verwendet werden. Das abgespeicherte Farb-RAM hat die Lade-Adresse $C400.

J – save Sprites – funktioniert wie das Laden von Sprites. Bei individueller Auswahl beendet <Pfeil links> die Zusammenstellung. Mit +/- lassen sich die schon aufgenommenen Sprites ansehen. Beim automatischen Aufnehmen werden Sprites so lange gespeichert, bis am Bildschirm nur noch leere Definitionen folgen.

Noch ein Tip:

Sollte eine Grafik beim Wandeln zu komplex sein, lassen sich auch die Funktionen des Logo-Painter (S. 4) verwenden. Speichern Sie dazu in einem der Formate Blazing Paddles, Koala-Painter oder Advanced OCP Studio. (gr)

Kurzinfo: MDG-Konvert

Programmart: Grafik-Konverter
Laden: LOAD"MDG-KONVERT",8
Starten: nach dem Laden RUN eingeben
Benötigte Blocks: 78 Blocks
Programmautor: Oliver Stiller

FMON 1541 - Diskettenmonitor für Spezialisten

Floppy-Bytes: Reifeprüfung

Direktzugriffe auf Spuren und Sektoren einer Diskette sind in Basic 2.0 kompliziert. »FMON« erforscht alle Floppy-Bytes und ändert sie nach Wunsch.

von M. Köhler und P. Baumann

Unser Diskettenmonitor erlaubt mit seinen Befehlen sogar, Assembler-Programme fürs Floppylaufwerk zu entwerfen. Laden Sie das Tool mit:

```
LOAD "FMON 1541",8,1
```

Geben Sie anschließend »NEW« ein. Mit »SYS 32768« startet man das Monitorprogramm. Hier eine Übersicht der komfortablen Anweisungen:

A (Direkt-Assembler)

Befehlsformat: A aaaa bbb ccccc

Der Befehl bbb mit dem Operanden ccccc wird an Adresse aaaa geschrieben. Alle Eingaben müssen hexadezimal erfolgen (ohne vorangestelltes <$>-Zeichen). Binärzahlen (mit %) und ASCII-Bytes (in Anführungsstrichen) sind erlaubt. Beispiele: `AND #%10101100 oder LDA #"A"`

Einzelne Bytes kann man mit vorangestelltem Punkt <.> eintippen (z.B. ».A9«). Bei Branch-Befehlen muß man die Zieladresse als zweistellige Hexzahl angeben.

D (Speicherbereiche disassemblieren)

Format: D aaaa, bbbb

Disassembliert den Speicherbereich aaaa bis bbbb. Ist keine der beiden Adreßangaben vorhanden, lautet die Anfangsadresse automatisch $0300. Möchten Sie einen Bytewert ändern, setzen Sie den Cursor an die entsprechende Stelle und tippen die neue Hexzahl ein.

X (Exit)

Das Basic 2.0 wird wieder eingeschaltet.

Z (Exit II)

Dieser Befehl wirkt wie die Tastenkombination <RUN/STOP RESTORE>.

G (Maschinenprogramm starten)

Format: G aaaa

Springt zu einer Unterroutine oder Assembler-Programm ab Adresse aaaa, das mit einem »RTS« ($60) abschließt.

M (Memory Dump)

Format: M aaaa, bbbb

Gibt den Speicherbereich aaaa bis bbbb mit 8 Byte pro Zeile (mit ASCII-Anzeige) auf dem Bildschirm aus. Fehlt bbbb, wirkt der Befehl wie die Anweisung »D«. Hat man aaaa nicht angegeben, listet das Programm den Bereich $0300 bis $03FF.

(Floppy-Adresse ändern)

Beispiel: #9. Die Geräteadresse der Diskettenstation wird auf »9« umgestellt.

> (Ändern der Druckeradresse)

Beispiel: >5. Der angeschlossene Drucker besitzt jetzt die Geräteadresse »5«.

@ (DOS-Befehle zur Floppy senden)

Format: @a: bbbbb

Die DOS-Anweisung a wird ans Laufwerk übermittelt. Beispiel: @s:datei (löscht das File »datei« auf der Diskette).

*** (Fehlerkanal auslesen)**

Die entsprechende DOS-Meldung erscheint auf dem Bildschirm. Das Floppy-Blinken wird abgestellt.

$ (Directory anzeigen)

Das Disketteninhaltsverzeichnis erscheint auf dem Bildschirm.

C (Kaltstart der Diskettenstation)

Entspricht einem Floppy-Reset. Achtung: Alle Floppy-Puffer werden gelöscht!

I (Initialize)

Die Diskettenstation wird neu initialisiert.

R (Block auf Diskette lesen)

Format: R tt ss

Lädt den Inhalt von Spur tt, Sektor ss in den Puffer 0 ($0300 bis $03FF). Die Zahlenangaben müssen hexadezimal erfolgen. Mit dem Befehl »M« kann man die Byte-Folge listen.

W (Block auf Diskette schreiben)

Format: W tt ss

»Read« und »Write« arbeiten mit Job-Befehlen. Eventuell auftretende Fehler werden in Adresse $0000 abgelegt.

J (Job-Befehl für Puffer 0 senden)

Format: J aa, bb, cc

Der Job-Befehl aa mit den Parametern bb und cc wird in die Speicheradresse $0000, 0006/0007 geschrieben.

P (schreibt ASCII-Text an gewünschte Stelle)

Format: P aaaa, ttttt

Ab Adresse aaaa werden die Textbytes ttttt abgelegt.

S (Speicherbereiche sichern)

Format: S »filename«, aaaa, bbbb

Der Bereich von aaaa bis bbbb wird in der sequentiellen Datei »filename« zwischengespeichert.

L (Speicherbereiche laden)

Format: L »filename«

Das Tool lädt die mit der »S«-Anweisung erzeugte Datei »filename«.

F (Bereiche füllen)

Format: F aaaa, bbbb, cc

Im Speicherbereich aaaa bis bbbb wird das Zeichen cc Byte für Byte abgelegt.

Q (Schreib-Lese-Kopf beschleunigen)

Format: Q oder Q0

»Q« erhöht die Geschwindigkeit, »Q0« stellt die Normalwerte wieder her.

N (Datei drucken)

Format:N »filename,X«

Der Inhalt des Files wird ausgedruckt. Für »X« muß der Buchstabe des entsprechenden Filetyps angegeben werden. (bl)

Kurzinfo: FMON 1541

Programmart: Diskettenmonitor
Laden: LOAD "FMON 1541",8
Starten: »NEW« eingeben, dann »SYS 32768«
Besonderheiten: erlaubt die gewohnte 6510-Assembler-Programmierart
Benötigte Blocks: 21
Programmautoren: M. Köhler/P. Baumann

Lebendig begraben

Unscratch – Dateien retten

Nichts, was der Computer macht, ist endgültig – auch wenn's auf den ersten Blick so aussieht. Gelöschte Files auf einer Diskette sind nicht wirklich verloren: »Unscratch« holt sie (Gott sei Dank) zurück.

von Nikolaus M. Heusler

Es ist zum Verzweifeln: Da hat man eine wertvolle Datei oder ein Superprogramm versehentlich auf der Diskette gelöscht. Eigentlich war ein ganz anderes File gemeint, aber leider hatte es einen ähnlichen Namen wie das gelöschte...

Keine Panik! Wenn Sie die Diskette nicht noch zusätzlich »hart« formatiert haben (mit ID), ist dies kein Beinbruch. Besitzer einer Floppy 1541 können z.B. Dateien vorsorglich vor der SCRATCH-Anweisung schützen (durch unser Programm »File Lock« im 64'er-Sonderheft 33). Wenn's trotzdem passiert, hilft Ihnen »Unscratch« aus der Patsche.

Bei der DOS-Anweisung zum Löschen einzelner Dateien

```
OPEN 15,8,15
PRINT#15,"S: Dateiname"
CLOSE 15
```

geschieht den Datenblocks auf der Diskette nicht viel. Das File wird lediglich im Disketteninhaltsverzeichnis als gelöscht (DEL) gekennzeichnet. Die Blöcke sind zur freien Verwendung wieder freigegeben. Sofern Sie nach dem versehentlichen Löschen kein anderes Programm oder eine Datei auf Diskette gespeichert haben, sollten Sie unmittelbar nach dem Unglück das Utility laden:

```
LOAD "UNSCRATCH",8
```

und mit RUN starten.

Wir versprechen Ihnen, daß Sie damit das verloren geglaubte File unversehrt zurückerhalten. Zur Bedienung benötigen Sie keine Programmierkenntnisse.

Auf dem Bildschirm erscheint die Aufforderung, die Diskette mit der gelöschten Datei einzulegen. Danach drücken Sie eine beliebige Taste. Der C64 liest nun das Directory dieser Diskette. Tritt ein Fehler auf, meldet sich der Computer mit einem entsprechenden Hinweis. Nach Tastendruck kann man das Programm erneut starten. Sämtliche denkbaren Diskettenfehler werden abgefangen und angezeigt.



▲ [1] Der entsprechende Filetyp muß eingetragen werden

Wichtiger Hinweis, wenn Sie mehrere Laufwerke haben: Nach dem Start stellt sich »Unscratch« auf das Laufwerk ein, das zuletzt angesprochen wurde. Normalerweise ist es das Gerät, von dem »Unscratch« geladen wurde. Ist das Programm im Zweifel, setzt es automatisch die Geräteadresse 8 (Laufwerk 0). Der C64 durchsucht jetzt das Inhaltsverzeichnis nach Dateien mit der Löschkennung »DEL« ($80). Wird er fündig, bringt das Programm den entsprechenden Filenamen auf den Bildschirm und fragt, ob es erneuert werden soll. Hat man sich für eine Restaurierung entschieden, folgt die Frage nach dem ursprünglichen File-Typ (s. Abb.):

<P>: Programm (PRG)
<S>: sequentielle Datei (SEQ)
<U>: User-File (USR)
<R>: relative Datei (REL)
<A>: Abbruch

Mit der zuletztgenannten Taste kann der Suchvorgang abgebrochen werden, um mit einer anderen Diskette weiterzumachen.

»Unscratch« ersetzt die Löschkennung $80 (DEL) mit dem Byte für die entsprechende Dateiart (Tabelle) und verfolgt die Blockverbindung, bis die Datei vollständig wiederhergestellt ist. Das Directory wird von oben nach unten gescannt, wie die Einträge im Blockbelegungsplan (BAM) verzeichnet sind und stellt nacheinander jedes gelöschte File zur Diskussion, ob es erneuert werden soll.

Wurde mindestens eine Datei restauriert, muß auch die Organisationsstruktur der Diskette neu vermerkt werden. Die Datenblöcke, in denen das renovierte File steht, müssen als belegt gekennzeichnet werden: Das Programm führt den DOS-Befehl »Validate« aus. Bei umfangreichen Directories kann das schon einige Minuten dauern. Anschließend können Sie wieder mit der Diskette und der erneuerten Datei arbeiten, als sei nichts geschehen.

Es kann vorkommen, daß man beim Wiederherstellen eines Files versehentlich den falschen Dateityp angegeben hat (statt »PRG« z.B. »SEQ«). Auch das ist kein Grund zur Beunruhigung. Löschen Sie die Datei sofort wieder, und stellen Sie sie mit »Unscratch« unter Angabe der richtigen Filebezeichnung wieder her. Beachten Sie außerdem, daß das Utility nur dann korrekt arbeitet, wenn nach dem Löschen einer Datei in der Zwischenzeit nichts anderes auf die Diskette gespeichert wurde. Da die Blöcke des gelöschten Programms wieder freigegeben wurden, kann es passieren, daß diese vom neu gespeicherten Programm überschrieben werden. In diesem Fall ist die ursprüngliche Datei allerdings unwiderruflich verloren. Das Programm wurde von uns mit den Laufwerken 1541-C, 1541-II und der Floppy 1571 im C128D getestet. Dabei traten keine Probleme auf.

Wir wollen hoffen, daß Sie künftig nicht zu viele Rettungsaktionen starten müssen, um versehentlich gelöschte Files wieder zum Leben zu erwecken – aber es ist vorgesorgt. (bl)

Dateiarten von Disketten-Files

Kennungsbyte	File-Typ
$80 (128)	DEL (gelöschte Datei, wird nicht im Directory angezeigt)
$81 (129)	SEQ (sequentielles File)
$82 (130)	PRG (Programmdatei)
$83 (131)	USR (User = Direktzugriffsdatei)
$84 (132)	REL (relativer Dateityp)

▲ Alle Filetypen der Floppy auf einen Blick

Kurzinfo: Unscratch

Programmart: Utility zum Wiederherstellen gelöschter Files
Laden: LOAD "UNSCRATCH",8
Starten: nach dem Laden mit RUN
Besonderheiten: kann alle gelöschten Dateiarten restaurieren
Benötigte Blocks: 5
Programmautor: Nikolaus M. Heusler

File-Compactor – Programme halbiert

Geschrumpfte Bytes

Auf jeder Diskette ist der Speicherplatz zu knapp. Unser Packer reduziert den Programmcode um mehr als 50 Prozent, ohne Qualitätsverlust!

von Arndt Brenschede

Die Zeit, in der man aus Platzgründen nur ein oder zwei Programme auf einer Diskette unterbrachte, können Sie vergessen: Unser Packer schafft Platz auf Ihren Disketten. Damit wird wirklich jedes Programm erheblich gekürzt. Effizient arbeitet der »File-Compactor« allerdings nur



[1] Ein Disketteninhaltverzeichnis vor...

mit Dateien, die länger als 25 Blöcke auf Diskette sind. Gerade in Basic-Programmen findet man viele Leerzeichen, Doppelpunkte, $00-Bytes oder häufig wiederkehrende Buchstaben (z.B. »E«). Das Utility codiert Zeichen, die sich oft wiederholen, mit einem 3 Bit langen Code. Je mehr im Programm bestimmte Zeichen dominieren, desto geringer ist der durchschnittliche Informationswert. Solche Files lassen sich erheblich kürzen. Selbstverständlich muß das Programm dabei für die seltener auftretenden Bytes längere Codes definieren (8 Bit und mehr), damit die Codierung eindeutig bleibt.

Der Packer wird wie jedes Basic-Programm geladen:

```
LOAD "FILE-COMPACTOR",8
```

Wenn Sie jetzt mit RUN starten, erscheint die Einschaltmeldung (als Hinweis, daß das Utility aktiviert ist):

```
AUFRUF: SYS 51000
UEBERLANGE PRG'S
SPEICHERN: SYS 50965
```

Das Programm im Visier muß ab Basic-Start ($0801 oder 2049) in den Speicher geladen werden. Eine Einschränkung besteht: Der gesamte Programmcode darf nicht mehr als 48 639 Byte umfassen.

Wenn sich nach dem Laden der Cursor wieder mit »READY« meldet, geben Sie den Startschuß:

```
SYS 51000
```

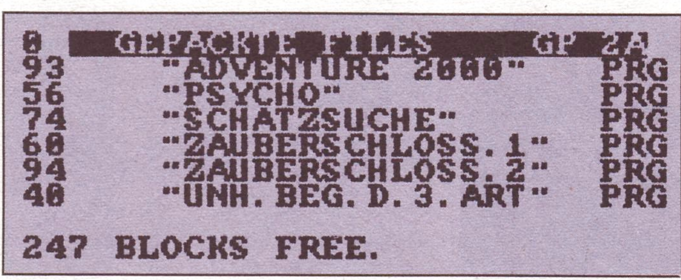
Je nach Umfang der Programmdatei dauert der Vorgang eine gewisse Zeit. Ist der Packer fertig, meldet er die prozentuale Verkürzung des Programms, z.B.:

```
REST : %68.3724026
```

Dies bedeutet, daß der »Filecompactor« das Basic-Programm um etwa ein Drittel gekürzt hat. Geben Sie nun den LIST-Befehl ein: Das Basic-Programm hat sich enorm verwandelt. Es erscheint nur noch eine Basic-Zeile:

```
43210 SYS2065 FCC
```

Das gepackte Basic-Programm läßt sich jetzt wie üblich speichern (mit SAVE). Vor den eigentlichen Programmcode schreibt der Packer eine Unterroutine in Maschinensprache, die fürs Entpacken zuständig ist. Sonst wäre das Programm beim späteren Laden nicht lauffähig: Der komprimierte



[2] ...und nach der Behandlung mit »File-Compactor«

Code wird erneut in ein lesbares Programm umgewandelt. Was das Utility leistet, sehen Sie an den Directory-Ausdrucken vor und nach dem Komprimieren (Abb. 1 und 2).

Der Packer durchsucht das gesamte Programm nach gleichen Byte-Werten. Wurde eine solche Folge gefunden, schreibt das Utility ein Merkzeichen ($BF oder $CF) an die aktuelle Stelle, anschließend den Code des Zeichens und, wieviele es sind. Nach dem Entpacken (Start mit RUN) wird die Basic-Datei Byte für Byte im entsprechenden Speicherbereich wieder installiert, exakt so, wie sie vor dem Packen im Basic-Speicher vorhanden war.

Sollen Programme gekürzt werden, die nicht in den Basic-Speicher passen bzw. sich bis unters ROM ausdehnen, können beim Speichern Probleme auftreten. Dafür bietet das Utility eine modifizierte SAVE-Routine. Verwenden Sie dazu folgende SYS-Anweisung, die Sie im Direktmodus eingeben müssen:

```
SYS 50965 "Filename"
```

Die Geräteadresse »8« (für die Floppystation) wird automatisch eingesetzt.

Sie werden staunen, wieviel zusätzlicher Speicherplatz plötzlich auf Ihren Disketten frei wird, wenn Sie den »File-Compactor« intensiv einsetzen. (bl)

Kurzinfo: File-Compactor

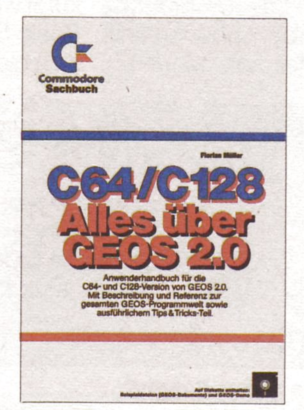
Programmart: Utility
Laden: LOAD "FILE-COMPACTOR",8
Starten: nach dem Laden mit RUN
Besonderheiten: arbeitet nur mit Programmen ab Basic-Start ($0801) zusammen
Benötigte Blocks: 7
Programmautor: Arndt Brenschede

C64/C128 Bücher und Bookware

Superkraftstoff z



H. Withöft/A. Draheim
64'er – Großer Einsteiger-Kurs
Schritt für Schritt werdet ihr in die Welt eures neuen Computers eingeführt. Vom Auspacken und Anschließen über Basic-Programmierung bis zu PEEK- und POKE-Befehlen. Wenn ihr auf der letzten Seite angekommen seid, habt ihr auch euren 64er im Griff.
1988, 236 Seiten,
inkl. Diskette
ISBN 3-890**90-668**-0
DM 29,90



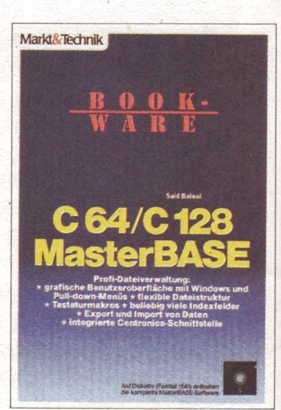
F. Müller
C64/C128 – Alles über GEOS 2.0
Das gesamte GEOS-2.0-Paket wird beschrieben, wobei den Umsteigern von früheren Versionen und Büchern ein spezieller Teil gewidmet ist. Mit Referenzteil zum Nachschlagen und umfangreichen Tips und Tricks.
Für alle C64-Anwender, die GEOS 2.0 perfekt beherrschen wollen.
1989, 432 Seiten,
inkl. Diskette
ISBN 3-890**90-808**-X
DM 59,–



F. Müller
C64 – Tips, Tricks und Tools
Eine wertvolle Sammlung toller Kniffe, um die Anwendung und Programmierung des 64ers noch effektiver zu gestalten und die Kenntnisse über euren Computer zu vertiefen. Leicht verständlich für den Einsteiger und immer wieder anregend für den Insider.
1988, 439 Seiten,
inkl. Diskette
ISBN 3-890**90-499**-8
DM 59,–



A. Seibert
C64/C128 Spielend Basic lernen
Dieser Basic-Grundkurs wendet sich besonders an Einsteiger und vermittelt alle nötigen Kenntnisse, um Basic-Programme schreiben zu können.
Alle Spiele sind als Listing im Buch abgedruckt und auf der Diskette enthalten. Sie brauchen einen C64 bzw. C128 (64er-Modus) mit einer Floppy 1541/70/71.
1989, 209 Seiten,
inkl. Diskette
ISBN 3-890**90-701**-6
DM 39,–



S. Baloui
C64/C128 MasterBase
Profi-Software zum Buchpreis: eine Dateiverwaltung für hohe Ansprüche und mit bequemer Benutzeroberfläche: Pull-down-Menüs, Windows, Makros und Indexfelder orientieren sich an professionellen Vorbildern. So lassen sich Adressen, Ton-Kassetten, Video-Sammlungen oder ähnliche Datenmengen problemlos verwalten.
1988, 155 Seiten,
inkl. Diskette
ISBN 3-890**90-583**-8
DM 59,–*

...um Normalpreis.

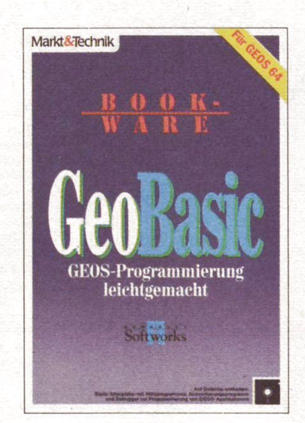
Wo gibt's denn sowas? Bei Markt&Technik. Eure Supertankstelle für Bücher und Bookware. 12mal neuer Profitreibstoff für Eure C64er und C128er. Zum Normalpreis. Da könnt Ihr Gas geben!

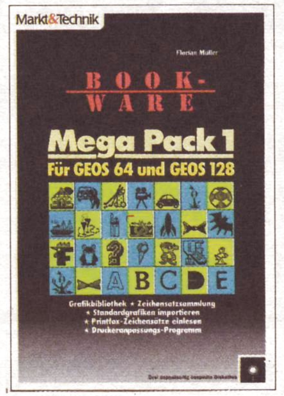
W. Knupe/H.-J. Ciprina
R. Bonse/V. Goehrke
MegaAssembler
Ein komplettes Entwicklungspaket, um lauffähige GEOS-Programme zu erzeugen. Umfangreiche und leistungsstarke Makros erleichtern die Arbeit. Alle im Buch beschriebenen Listings und die GEOS-Symboltabelle sind auf der Diskette gespeichert.
1990, 431 Seiten,
inkl. Diskette
ISBN 3-89090-247-2
DM 89,–*

F. Müller
Mega Pack 1
Profi-Software zum Buchpreis: eine nützliche Ergänzung, die euer GEOS-System noch vielseitiger macht. Über Pull-down-Menüs werden eine Grafikbibliothek oder Zeichensätze geladen, konvertiert und alles optimal dem Drucker angepaßt.
1989, 160 Seiten,
inkl. 3 Disketten
ISBN 3-89090-772-5
DM 59,–*

F. Müller
Mega Pack 2
Profi-Software zum Buchpreis: eine Diskart-Grafikbibliothek mit mehreren hundert Grafiken im Geo-Paint-Format. Mit Zeichensätzen und kunstvollen Symbolen, die euren »Drucksachen« eine individuelle Note verleihen. Ihr müßt nur noch aussuchen, markieren, einkleben – fertig.
1989, 177 Seiten,
inkl. 3 Disketten
ISBN 3-89090-350-9
DM 59,–*

Berkeley Softworks
GeoBasic
GeoBasic enthält neben einem schnellen Editor über 100 Befehle und Funktionen, die die Fähigkeiten von GEOS ausnutzen. Mit dem Konvertierungsprogramm können C64-Basic-Programme übernommen werden, und der eingebaute Debugger hilft bei der Fehlersuche.
1990, 212 Seiten,
inkl. Diskette
ISBN 3-89090-245-6
DM 89,–*





*unverbindliche Preisempfehlung







U. Gerlach
Hardware-Basteleien zum C64/C128
Centronic-Druckertreiber, Floppy-Speeder im Eigenbau? Hier findet ihr die leichtverständliche Einführung in die digitale Schalttechnik und die Schnittstellen des C64/C128. Mit vielen Platinen-Layouts und genauen Bauanleitungen, Stücklisten und Bezugsquellen für viele interessante Erweiterungen und nützlichen Zusätzen. Doityourself!
1987, 294 Seiten,
inkl. Diskette
ISBN 3-89090-389-4
DM 49,–

A. Seibert
C64'er-Spielesammlung, Band 3
Beste Unterhaltung und ausgiebiger Spaß vermitteln diese 12 spannenden und reizvollen Spiele aus der Welt der Agenten, Bösewichter und Zauberer. Ein Schnellader auf der Diskette verkürzt die Ladezeiten auf einen Bruchteil der ursprünglichen Ladezeit.
1988, 103 Seiten,
inkl. Diskette
ISBN 3-89090-596-X
DM 39,–*

A. Woerrlein
64'er-Spielesammlung, Band 4
20 ausgewählte Spiele, die Geschicklichkeit und Fingerspitzengefühl verlangen – und die noch besser auf die Bedürfnisse eines echten Spiele-Freaks zugeschnitten sind. Kommt mit in eine Welt, die vor euch noch niemand zu Gesicht bekommen hat.
1988, 80 Seiten,
inkl. Diskette
ISBN 3-89090-703-2
DM 39,–*

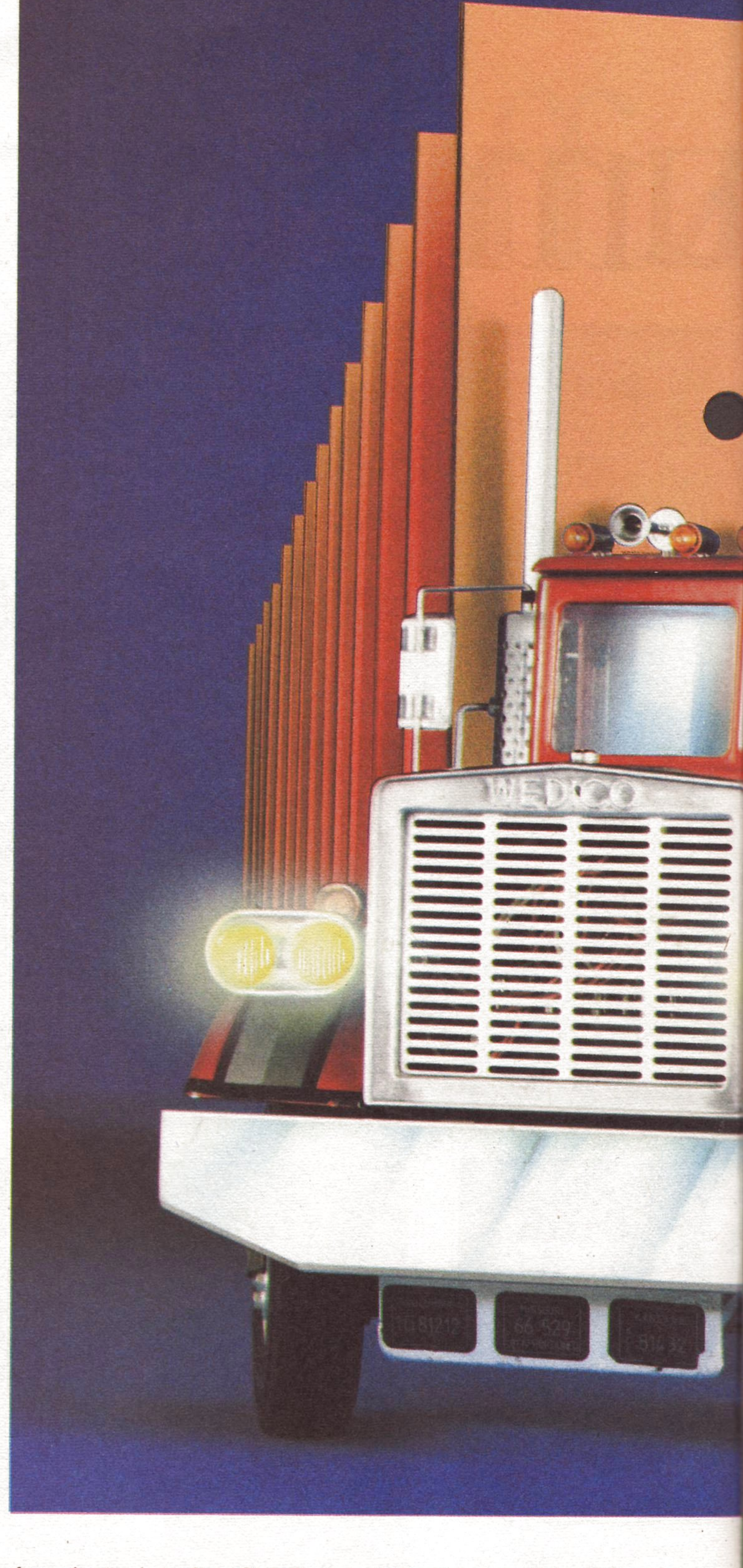
Markt&Technik-Bücher und -Software erhalten Sie bei Ihrem Buchhändler, in Computerfachgeschäften und Fachabteilungen der Warenhäuser.



1306/010

Tips & Tricks-Factory – frisch ab Werk

Heiße Fracht



Aufgetankt, vollbeladen mit hochkarätigen Tips und Tools, macht sich unser Trucker auf den Weg, um den C-64-Fans wertvolle (Soft-)Ware frei Haus zu liefern. Staunen Sie, was sich alles auf der Ladefläche befindet!

Autostart mit Revanche

Dieses Utility erzeugt nicht nur selbststartende Basic- und Assembler-Programme. Es setzt zusätzlich die Taste <RUN/STOP> außer Betrieb und installiert einen Kopierschutz. Mit einem einzigen Befehl läßt sich der gesamte Vorgang auch wieder rückgängig machen.

Laden Sie das Utility mit:

```
LOAD "PBASIC",8
```

Nach dem Start mit RUN erfolgt die Einschaltmeldung. Das Basic 2.0 wurde um drei Anweisungen erweitert, die mit dem Zeichen <!> eingeleitet werden. Hier die Erläuterung:

!SAVE »Programmname«,8,1

Speichert ein Basic- oder Assembler-Programm (mit Basic-Startzeile, z.B. »SYS 2062«) in veränderter Form. Bei künftigem Laden startet es automatisch. Die RUN/STOP-Taste ist verriegelt, das Programm läßt sich also mit gebräuchlichen Mitteln nicht unterbrechen. Auch das Kopieren funktioniert nur noch mit bestimmten Copy-Tools. Die neue Programmdatei darf nach dieser Änderung nur noch »absolut«, d.h. mit dem Zusatz »,8,1« geladen werden.

!LOAD »Programmname«,8,1

Oft will man solche geschützten Basic-Programme nach einiger Zeit erweitern oder verbessern. Im neuen Format ist dies nicht möglich: Man muß alle Schutz- und Autostart-Einrichtungen wieder rückgängig machen. Mit der !LOAD-Anweisung knacken Sie alle von »PBasic« bearbeiteten Files und machen Sie editierfähig.

!NEW

Tritt bei der Arbeit mit den erweiterten Anweisungen ein Fehler auf (Floppy/Computer), läßt sich dieser mit dem neuen NEW-Befehl korrigieren. Alle wichtigen Vektoren des Betriebssystems werden richtiggestellt. Das Programm mit Autostart kann gelistet und bearbeitet werden.

Beachten Sie, daß die drei Erweiterungsbefehle nur im Direktmodus eingegeben werden dürfen. (Jan Kusch/bl)

Gesammelte Programmierhilfen

Zugegeben, die Basic-Erweiterung »Exbasic Level II« (64'er-Sonderheft 62) ist eine der besten, die je für den C64 veröffentlicht wurden – aber sie reduziert den knappen Basic-



Speicher des C 64 auf 30 KByte. Außerdem benötigt man die 70 neuen Befehle selten gleichzeitig. »Tool-Kit« bietet quasi einen Extrakt dieser Erweiterung: Acht leistungsfähige Befehle und zwei zusätzliche Funktionen bieten jedem Basic-Programmierer wertvolle Hilfe.

Laden Sie das Tool von der Diskette mit:

```
LOAD "TOOL-KIT",8
```

Folgende Befehle stehen ab sofort zur Verfügung:

!AUTO Startzeile, Schrittweite

Dieser Befehl macht Ihnen das Eintippen der Zeilennummern überflüssig. »Startzeile« bedeutet die Zahl der Anfangszeile, bei der Sie mit dem Programmieren beginnen. Als »Schrittweite« trägt man den Abstand zur nächsten Zeilennummer ein. Sie ist auf den Wert »255« limitiert. Wenn die Anweisung »AUTO« eingetippt ist, erscheint nach Druck auf die Tastenkombination <CBM F7> die erste Zeilennummer, jede weitere Betätigung dieser Kombination gibt die nächste Zeilennummer aus, erhöht um »Schrittweite«. Mit den Tastenkombinationen <CBM F1> oder <RUN/STOP RESTORE> läßt sich die Auto-Funktion wieder abschalten.

!DUMP

Das Programm listet alle Variablen eines Basic-Programms mit deren aktuellem Inhalt. Ausnahme: Indizierte Variablen! Mit <CTRL> verlangsamen Sie die Bildschirmausgabe, mit <STOP> wird sie abgebrochen.

!FIND

Man muß zwischen der Suche nach Zeichenketten (Strings, z.B. !FIND »TEST«, !FIND A$) oder nach Befehlen oder Variablen (z.B. !FIND GOTO1000, !FIND POKE) unterscheiden. Auf dem Bildschirm erscheinen die kompletten Basic-Zeilen, die den Suchbegriff enthalten. Auch hier kann man die Listenausgabe mit <CTRL> verlangsamen.

!KILL Angangszeile – Endzeile

Löscht einzelne Zeilen oder ganze Blöcke. Beispiele:
!KILL 100-200: alle Zeilen ab 100 bis inkl. 200,
!KILL -300: alle Zeilen ab Programmanfang bis inkl. Zeile 300,
!KILL 1800- : alle Zeilen ab inkl. 1800 bis Programmende.

Die Nummer der ersten Zeile muß kleiner sein als die der Endzeile. Außerdem muß die Anfangszeilennummer im Programm unbedingt existieren, sonst gibt's einen »Syntax Error«. Bei umfangreichen Zeilenblöcken, die zu löschen sind, kann es allerdings eine Weile dauern. Mit <F7> brechen Sie diese Funktion ab.

!LOAD

Damit hängen Sie an ein Basic-Programm im Speicher ein weiteres:

```
!LOAD "Programm 2",8
```

Die nachgeladene Programmdatei muß mit einer höheren Zeilennummer beginnen, als sie die letzte des Programms im Speicher besitzt. Eventuell muß vorher die !RENUM-Funktion ausgeführt werden. Bei identischen Zeilennummern beider Programme können sich sonst beim Anpassen der Sprungbefehle Fehler einstellen.

!RENUM

Damit verteilen Sie neue Zeilennummern an Ihr Programm, wahlweise mit gewünschter Schrittweite. Sämtliche Nummern hinter Sprungbefehlen (GOTO, GOSUB, RUN usw.) werden berücksichtigt und neu berechnet. Der Befehl bietet mehrere Möglichkeiten:

!RENUM: Ohne Parameter wird das Programm in Zehnerschritten durchnumeriert, beginnend mit Zeile 10.

!RENUM X,Y: »X« ist die erste neue Zeile, »Y« die Schrittweite.

!RENUM X,Y,Z: Erst ab der Zeile mit dem Wert »Z« wird neu numeriert. »Z« muß kleiner sein als »X«.

!TRACE

Schritt für Schritt läßt sich ein Programm nachverfolgen. Geben Sie zuerst diese Anweisung im Direktmodus ein und starten Sie ein Basic-Programm mit RUN. In den beiden oberen Bildschirmzeilen wird die Zeile gelistet, die das Programm gerade bearbeitet. Der aktuelle Befehl erscheint revers. Benutzen Sie folgende Tasten, um die TRACE-Funktion zu steuern:
<F1>: schneller Testlauf,
<F7>: es wird nur immer ein Befehl abgearbeitet.

Wenn Sie im Schnell-Trace-Modus (<F1>) die STOP-Taste drücken, läßt sich diese Funktion abbrechen.

!OFF

Deaktiviert den TRACE-Modus.

Alle hier aufgeführten Befehle lassen sich auch innerhalb eines Programms benutzen. Sinnvoll sind aber nur !DUMP, !TRACE und !OFF. Zwei zusätzliche Funktionen sind in »Tool-

Kit« integriert, die nicht mit einem Befehl aktiviert werden:
- Findet der C64 während des Programmablaufs einen Fehler, wird die betreffende Zeile gelistet. Der Cursor steht darunter.
- Eine Programmliste (Befehl LIST) läßt sich mit der Taste <F7> anhalten, ein weiterer Druck auf <SPACE> entriegelt diese Sperre.
- Ein gestopptes Listing kann nur mit <F1> abgebrochen werden, nicht mit der STOP-Taste. (Herbert Kunz/bl)

Tastenmanipulationen

Ein ruhiges Leben führten bisher die Funktionstasten <F1> bis <F8>. Ab sofort gibt's Arbeit:

```
LOAD "KEYS",8
```

Nach dem Start mit RUN meldet sich der C64 zwar nur mit einem nichtssagenden »READY«, trotzdem hat sich etwas geändert: Das Basic 2.0 wurde um die beiden Befehle »!KEY« und »!DISPLAY« erweitert. Außerdem kann man die wichtigsten Basic-Befehle durch gleichzeitigen Druck auf <CTRL> und eine bestimmte Buchstabentaste auf dem Bildschirm erscheinen lassen (Tabelle 1).

Insgesamt stehen zwölf (statt acht) Funktionstasten zur Verfügung: die normale Belegung (F1, F3, F5, F7) sowie die mit den Tasten <SHIFT> (F2, F4, F6, F8) und <CBM> (F9, F10, F11, F12). Mit folgender Anweisung werden die Tasten mit beliebigem Text ausgestattet:

!KEY Nummer, »Text«. Für »Nummer« muß hier eine Zahl zwischen »1« und »12« stehen, die Anzahl der Zeichen für »Text« darf allerdings nicht größer als »10« sein. Sollen Anführungszeichen im Textstring vorkommen, müssen Sie dafür das Hochkomma eingeben. Wenn die der F-Tastenbefehl sofort ausgeführt werden soll, müssen Sie das Zeichen »Engl. Pfund« <£> anhängen. Damit simuliert das Programm die RETURN-Taste.

!DISPLAY löscht den Bildschirm und zeigt die aktuelle Belegung der zwölf Funktionstasten auf einen Blick. Man kann den Text dann editieren. Empfehlenswert ist es, die DISPLAY-Anweisung ebenfalls auf eine Funktionstaste zu legen.

(Klaus Russell/bl)

ESF – sequentielle Files editieren

SEQ-Dateien besitzen einen Nachteil: Sie lassen sich nur von Programmen laden, die eine entsprechende Leseroutine besitzen. In der Regel sind das dieselben Hauptprogramme, die solche Files erzeugen. Ist so ein Master-Programm allerdings nicht zur Hand, bleibt dem interessierten Computerfan der Inhalt sequentieller Dateien verborgen: Der normale Ladebefehl (LOAD »Filename«,8) bringt nur die Fehlermeldung »File not found«. Einziger Ausweg: eine eigene Routine programmieren – oder »ESF« benutzen:

```
LOAD "ESF",8
```

Nach dem Start mit RUN lädt der Computer die Routine »DIR« zur Directory-Ausgabe, die sich mit der Taste <D> aktivieren läßt. »X« beendet das Programm, mit <M> ruft man das Arbeitsmenü auf (Abb.1). Der Computer verlangt jetzt die



[1] Sequentielle Dateien komfortabel editieren: das Hauptmenü von »ESF«

Eingabe des Filenamens der squentiellen Datei, die Sie verändern oder sich nur ansehen möchten. Als Demo-File finden Sie auf der beiliegenden Diskette »SYS«, eine Liste interessanter Speicheradressen im C64. Das erste Datenfeld bezieht sich auf die Adresse (hexadezimal/dezimal), das zweite gibt Auskunft über deren Funktion.

Tastenbelegung mit KEYS

So ändern Sie die voreingestellte Belegung: Man entnimmt folgender Tabelle den Tasten-Codewert und addiert »49579«. Anschließend subtrahiert man vom Token-Wert die Zahl »127« und POKEt das Ergebnis in die berechnete Adresse.
Folgende Tasten müssen zusammen mit der Taste <CTRL> gedrückt werden:

Taste	Code (PEEK(203))	Befehl
< + >	40	SIN
<->	43	COS
< >	48	TAN
<F1>	4	RUN
<F3>	5	LIST
<F5>	6	LOAD
<F7>	3	SAVE
<W>	9	WAIT
<E>	14	END
<R>	17	RETURN
<T>	22	TO
<Y>	25	SYS
<U>	30	USR
<I>	33	INPUT
<O>	38	OPEN
<P>	41	POKE
<@>	46	CONT

Taste	Code (PEEK(203))	Befehl
<*>	49	SQR
<A>	10	ABS
<S>	13	STEP
<D>	18	DATA
<F>	21	FOR
<G>	26	GOTO
<H>	29	THEN
<J>	34	READ
<K>	37	RIGHT$
<L>	42	LEFT$
<:>	45	INT
<;>	50	LOG
<Z>	12	GOSUB
<X>	23	PEEK
<C>	20	CLOSE
<V>	31	VAL
<B>	28	RESTORE
<N>	39	NEXT
<M>	36	MID$
<,>	47	TAB(
<.>	44	SPC(
<CRSR abwärts>	7	EXP
<CRSR rechts>	2	CONT

Tabelle 1. Basic-Befehle, die mit »Keys« auf Tastendruck erzeugt werden

Der Bildschirm zeigt die Datensätze fortlaufend, wie sie in der SEQ-Datei abgelegt sind. Jedes Datenfeld (Bildschirmzeile) versieht das Programm mit einer reversen Nummer. Mit der Taste <SPACE> läßt sich die Ausgabe stoppen, mit <RETURN> fortsetzen. Ihr Werkzeug zur Bearbeitung der Datei:

<E> Datenfeld einfügen: Geben Sie die Nummer des davorstehenden Datenfelds ein. Dann können Sie den gewünschten Text fürs neue Datenfeld tippen. Da »ESF« mit einer GET-Abfrage arbeitet, dürfen alle Tastaturzeichen eingegeben werden, die beim INPUT-Befehl verboten sind.

<C> Datenfeld ändern: Nach Angabe der entsprechenden Datenfeldnummer muß man den Ersatztext eingeben.

<L> Datenfeld löschen: Das Datenfeld mit der gewünschten Nummer wird aus dem Datensatz entfernt.

<D> SEQ-Datei drucken: Damit bringt man die gesamte Datei über einen seriell angeschlossenen Drucker zu Papier. Die Druckroutine ab Listingzeile 1730 benutzt die Sekundäradresse »7«. Falls Sie für Ihren Drucker einen anderen Wert benötigen, können Sie dies in Zeile 1840 ändern.

<W> Weiter: Die geänderte Datei läßt sich mit einem beliebigen Filenamen erneut speichern. Falls die Datei bereits auf Diskette existiert, kann man sie nach einer Sicherheitsabfrage überschreiben. Das Utility wird erneut gestartet.

(G. Kluge/bl)

Bildschirminhalt sichern

Das Utility speichert jeden Textbildschirm des C64 mit entsprechendem Filenamen auf Diskette. Es berücksichtigt außer Bildschirminhalt (Adresse 1024 bis 2023) auch das beim Speichern aktuelle Farb-RAM (Adresse 55296 bis 56295).

Nach dem Laden des Programms mit:

```
LOAD "SCREENSAVER",8
```

und dem Start mit RUN wird das Hilfsprogramm initialisiert und gibt die Systemadressen fürs Speichern und Laden bekannt:

```
SYS 40738 "Bildname",8 (Screen speichern)
SYS 40833 "Bildname",8 (Screen laden)
```

Da man mit Eingaben im Direktmodus den Bildschirm zerstört, sollten Sie die beiden Anweisungen am besten in einem Programm verwenden. (Ralph Babel/bl)

Treppauf – treppab

Der Bildschirm scrollt eine Zeile nach oben, sobald der Cursor oder z.B. Ausgabetext die unterste Zeilenposition (Nr. 24) erreicht hat. Problematisch wird es, wenn man dieses Scrollen auslösen will, obwohl die unterste Zeile noch nicht erreicht ist. Dazu besitzt der C64 ebenfalls eine Betriebssystemroutine: **SYS 59626 ($E8EA)**. Damit scrollt der Bildschirm um eine Zeile nach oben. In die entgegengesetzte Richtung geht es mit: **SYS 59749 ($E965)**. Sollen mehr Zeilen bewegt werden, muß die SYS-Anweisung in eine FOR-NEXT-Schleife eingebunden werden. Unsere Programmbeispiele »SCROLL1« und »SCROLL2« zeigen im Listing, wie man vorgehen muß. Experimentieren Sie mit dieser Routine. Sie bietet interessante Effekte.

Floppyfehler ohne Rätsel

Leider fehlt dem Basic 2.0 eine Funktion zur Abfrage des Fehlerkanals, wenn das Diskettenlaufwerk per blinkender LED einen Fehler anzeigt. »Fehlermeldung« überwacht Situationen, die zu Floppyfehlern führen können: eine Datei laden, die auf Diskette nicht vorhanden ist, der Ladeversuch eines unerlaubten File-Typs (SEQ, USR) usw. Das Programm kann man mit:

```
LOAD "FEHLERMELDUNG",8
```

von der beiliegenden Diskette laden und mit RUN starten. Es läßt im freien RAM beliebig verschieben. Schleicht sich jetzt beim Betrieb mit der Diskettenstation ein Fehler ein, bringt es den Klartext der Fehlermeldung auf den Bildschirm (Sie wissen sofort, woran es haperte) und stellt das nervöse Blinken der Floppy-LED ab. (Georg Kramer/bl)

Hypra-Trilogie

HYPRA-LOAD

Kaffeepause! Daran denkt jeder C-64-Fan, der ein längeres Programm laden möchte. Aber: Mit **»Hypra-Load«** geht's fünfmal schneller! Laden Sie unseren Software-Ladespeeder mit:

```
LOAD "HYPRA-LOAD",8,1
```

Bei Erscheinen der Einschaltmeldung steht das Utility zur Verfügung. Laden Sie nun das gewünschte Basic- oder Assembler-Programm wie gewohnt. Der Bildschirm bleibt während des Ladens ausgeschaltet. Mit der Anweisung »POKE 1,55« läßt sich »Hypra-Load« deaktivieren.

(B. Schneider/K. Schramm/bl)

HYPRA-SAVE

Diese Ergänzung zu »Hypra-Load« speichert Programme drei- bis fünfmal schneller. Sie läßt sich wie ein Basic-Programm laden und mit RUN starten:

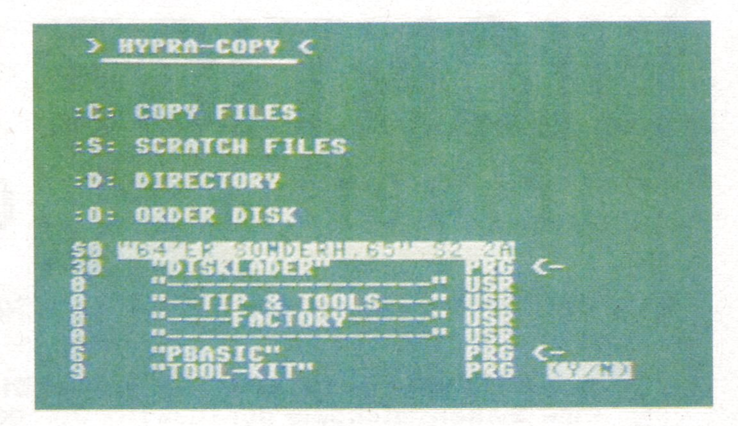
```
LOAD "HYPRA-SAVE",8
```

Auf dem Bildschirm erscheint die Anweisung, um das Utility zu aktivieren: SYS 365. Anschließend sollte man »NEW« eingeben, um den Basic-Speicher für andere Programme wieder freizumachen. Alle folgenden Speichervorgänge benutzen jetzt die schnellen Routinen von »Hypra-Save«.

(Martin Pfost/bl)

HYPRA-COPY

Mit diesem Utility kann man zwar kein Disketten-Backup ziehen, jedoch werden einzelne Files vier- bis fünfmal schneller auf eine andere Diskette übertragen. Achtung: Relative



[2] Mit <RETURN> überträgt »Hypra-Copy« alle ausgewählten Files auf die Zieldiskette

Dateien (Kennzeichen REL) oder Files, die größer als 232 Blocks auf Diskette sind, kann »Hypra-Copy« nicht bearbeiten!

Laden Sie das Programm von der beiliegenden Diskette:

```
LOAD "HYPRA-COPY",8
```

und starten Sie es mit RUN. Das Hauptmenü bietet folgende Auswahlpunkte:

<C> Copy Files: Auf dem Bildschirm erscheint das Directory. Das Programm stoppt nach jedem File-Eintrag und fragt,

So finden Sie die Programme auf der Diskette

DISKETTE SEITE 1

```
0  "64'ER SONDERH.65" S1 2A
30    "DISKLADER"           PRG   Seite 21
0     "------------------"  USR
0     "------GRAFIK------"  USR
0     "------------------"  USR
45    "LP3+(FULL)"          PRG   Seite 4
29    "LP3+(SHORT)"         PRG
90    "4.INNERSPACE/SD!"    PRG
9     "SCREEN      /X3+"    PRG
9     "CHAR        /X3+"    PRG
2     "EINBINDUNGS-BSP."    PRG
41    "PI.DISKETTE"         PRG
17    "DISKETTE    /X3+"    PRG
40    "PIC UNICORN     "    PRG
17    "UNICORN     /X3+"    PRG
40    "WERNER      MPIC"    PRG
17    "WERNER      /X3+"    PRG
0     "------------------"  USR
78    "MDG-KONVERT     "    PRG   Seite 8
0     "------------------"  USR
0     "-----FLOPPY-----"    USR

0     "------------------"  USR
21    "F-MON.1541"          PRG   Seite 9
0     "------------------"  USR
5     "UNSCRATCH"           PRG   Seite 10
0     "------------------"  USR
7     "FILE-COMPACTOR"      PRG   Seite 11
0     "------------------"  USR
0     "-PROGRAMMIEREN--"    USR
0     "------------------"  USR
5     "Z-PAGE"              PRG   Seite 24
1     "ROM TO RAM"          PRG
3     "ADRESSE 19"          PRG
2     "ADRESSE 199"         PRG
6     "ADRESSE 203"         PRG
3     "ADRESSEN 204/207"    PRG
3     "ADRESSEN 211/214"    PRG
2     "LADEN"               PRG
1     "SPEICHERN"           PRG
0     "------------------"  USR
6     "ECHTZEITUHR"         PRG   Seite 30
2     "SENDEN"              PRG

1     "EMPFANGEN"           PRG
0     "------------------"  USR
6     "RASTER-IRQ"          PRG   Seite 34
8     "SPRITE-IRQ"          PRG
10    "LIGHTP.-IRQ"         PRG
0     "------------------"  USR
0     "-----DRUCKER------"  USR
0     "------------------"  USR
8     "DRUCKER-BASIC"       PRG   Seite 48
4     "INTERFACE"           PRG
0     "------------------"  USR
0     "-----DISKETTE-----"  USR
0     "---BEIDSEITIG---"    USR
0     "-----BESPIELT-----"  USR
0     "------------------"  USR
96 BLOCKS FREE.

READY.
```

DISKETTE SEITE 2

```
0  "64'ER SONDERH.65" S2 2A
30    "DISKLADER"           PRG   Seite 21
0     "------------------"  USR
0     "--TIP & TOOLS---"    USR
0     "----FACTORY------"   USR
0     "------------------"  USR
6     "PBASIC"              PRG   Seite 14
9     "TOOL-KIT"            PRG
3     "KEYS"                PRG
0     "------------------"  USR
14    "ESF"                 PRG
1     "DIR"                 PRG
6     "SYS"                 SEQ
0     "------------------"  USR
2     "SCROLL1"             PRG
3     "SCROLL2"             PRG
0     "------------------"  USR
5     "HYPRA-LOAD"          PRG
6     "HYPRA-SAVE"          PRG
15    "HYPRA-COPY"          PRG
0     "------------------"  USR
1     "ESCAPE-TASTE"        PRG
3     "PAUSE"               PRG
9     "FEHLERMELDUNG"       PRG
0     "------------------"  USR
5     "AUTO-SAVE"           PRG
1     "SAVE-OBJ"            PRG
0     "------------------"  USR

22    "MENUE-MAKER"         PRG
19    "DISK FUELLER"        PRG
5     "SCREENSAVER"         PRG
5     "NEUES INPUT"         PRG
2     "BMC.EXE"             PRG
0     "------------------"  USR
13    "BKS 5.0 (49152)"     PRG   Seite 40
21    "BKS.WHAT 5.0"        PRG
4     "SYNTAX-TEST"         PRG
0     "------------------"  USR
20    "CROSS-REF 64"        PRG   Seite 43
2     "CROSS.DEMO"          PRG
1     "STARTADRESSE"        PRG
0     "------------------"  USR
0     "--DIE EINZEILER-"    USR
0     "-------VOM-------"   USR
0     "-TIPS & TRICKS--"    USR
0     "------POSTER------"  USR
0     "------------------"  USR
1     "OLD"                 PRG   Seite 2
1     "MERGE"               PRG   und
1     "DIRECTORY"           PRG   Seite 51
1     "RENUMBER"            PRG
1     "INPUT M.KOMMA"       PRG
1     "PRINT AT"            PRG
1     "FLOPPY ON/OFF"       PRG
1     "DRUCKER ON/OFF"      PRG
1     "FLOPPYSPEEDER"       PRG

1     "KOPIERSCHUTZ"        PRG
0     "------------------"  USR
0     "-----SUPER-------"   USR
0     "---20-ZEILER-----"   USR
0     "------------------"  USR
5     "ZEICHENSATZ-EDI"     PRG   Seite 46
9     "ZEICHENSATZDEMO"     PRG
0     "------------------"  USR
7     "MINITEXT V1.0"       PRG
0     "------------------"  USR
7     "EXTEND BASIC"        PRG
4     "EXTEND BASIC.MC"     PRG
0     "------------------"  USR
5     "FIND/REPL.BAS"       PRG
2     "FIND/REPLACE"        PRG
0     "------------------"  USR
6     "MICRO-WRITER+"       PRG
3     "MW.DEMO"             PRG
0     "------------------"  USR
2     "MEMSAVE.INSTALL"     PRG
1     "MEMSAVE 828"         PRG
0     "------------------"  USR
0     "-------ENDE-------"  USR
0     "------------------"  USR
370 BLOCKS FREE.

READY.
```

WICHTIGE HINWEISE

zur beiliegenden Diskette:

Aus den Erfahrungen der bisherigen Sonderhefte mit Diskette wollen wir ein paar Tips an Sie weitergeben:

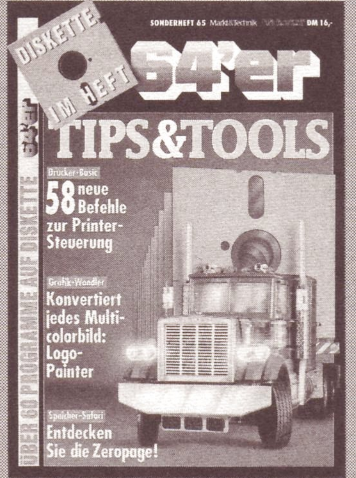
1 Bevor Sie mit den Programmen auf der Diskette arbeiten, sollten Sie unbedingt eine Sicherheitskopie der Diskette anlegen. Verwenden Sie dazu ein beliebiges Kopierprogramm, das eine komplette Diskettenseite dupliziert.

2 Auf der Originaldiskette ist wegen der umfangreichen Programme nur wenig Speicherplatz frei. Dies führt bei den Anwendungen, die Daten auf die Diskette speichern, zu Speicherplatz-Problemen. Kopieren Sie daher das Programm, mit dem Sie arbeiten wollen, mit einem File-Copy-Programm auf eine leere, formatierte Diskette und nutzen Sie diese als Arbeitsdiskette.

3 Die Rückseite der Originaldiskette ist schreibgeschützt. Wenn Sie auf dieser Seite speichern wollen, müssen Sie vorher mit einem Diskettenlocher eine Kerbe an der linken oberen Seite der Diskette anbringen, um den Schreibschutz zu entfernen. Probleme lassen sich von vornherein vermeiden, wenn Sie die Hinweise unter Punkt 2 beachten.

ALLE PROGRAMME

aus diesem Heft



HIER

...leitung ...he Rückseite ▶

IMPRESSUM

Herausgeber: Carl-Franz von Quadt, Otmar Weber

Redaktionsdirektor: Dr. Manfred Gindle

Chefredakteur: Georg Klinge - verantwortlich für den redaktionellen Teil
Stellv. Chefredakteur: Arnd Wängler
Textchef: Jens Maasberg
Produktion: Andrea Pfliegensdörfer
Redaktion: Harald Beiler (bl), Herbert Großer (gr)
Mitarbeiter dieser Ausgabe: Nikolaus Heusler
Redaktionsassistenz: Sylvia Derenthal, Diana Moser (089/4613202)
Telefax: 089/4613-5001
Alle Artikel sind mit dem Kurzzeichen des Redakteurs und/oder mit dem Namen des Autors/Mitarbeiters gekennzeichnet

Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten worden sein, muß dies angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in von der Markt&Technik Verlag AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträger. Mit der Einsendung von Bauanleitungen gibt der Einsender die Zustimmung zum Abdruck in von Markt&Technik Verlag AG verlegten Publikationen. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Verlagsleitung: Wolfram Höfler
Operation Manager: Michael Koeppe

Art-director: Friedemann Porscha
Layout: Marian Schwarz
Bildredaktion: Roland Müller (Fotografie); Ewald Standke, Norbert Raab (Spritzgrafik); Werner Nienstedt (Computergrafik)

Anzeigendirektion: Jens Berendsen
Anzeigenleitung: Philipp Schiede (399) - verantwortlich für die Anzeigen
Telefax: 089/4613-775

Anzeigenverwaltung und Disposition: Chris Mark (421)

Auslandsrepräsentation:
Auslandsniederlassungen:
Schweiz: Markt&Technik Vertriebs AG, Kollerstr. 37, CH-6300 Zug, Tel. 042-440550/660, Telefax 042-415770
USA: M&T Publishing Inc.; 501 Galveston Drive Redwood City, CA 94063, Telefon: (415) 366-3600, Telex 752-351
Österreich: Markt&Technik Ges. mbH, Große Neugasse 28, A 1040-Wien Telefon: 0222/5871393, Telex: 047-132532
Anzeigen-Auslandsvertretung:
England: F. A. Smyth&Associates Limited, 23a, Aylmer Parade, London, N2 OPQ. Telefon: 0044/1/3405058, Telefax: 0044/1/3419602
Israel: Baruch Schaefer, Haeskel-Str. 12, 58348 Holon, Israel, Tel. 00972-3-5562256
Taiwan: Aim International Inc., 4F.-1, No. 200, Sec. 3, Hsin-l Rd.; Taipei, Taiwan, R.O.C., Tel. 00886-2-7548631, -7548633, Fax 00886-2-7548710
Korea: Young Media Inc., C.P.O. Box: 6113, Seoul/Korea, Tel. 0082-2-7564819, /-7742759, Fax 0082-7575789
USA: M&T Publishing Inc.; 501 Galveston Drive Redwood City, CA 94063, Telefon: (415) 366-3600, Telex 752-351

Vertriebsdirektor: Uwe W. Hagen
Vertriebsmarketing: Petra Schlichthärle (703)

Vertrieb Handel: Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: ip Internationale Presse, Ludwigstraße 26, 7000 Stuttgart 1, Tel. 0711/619660

Bezugsmöglichkeiten: Leser-Service: Telefon (089) 4613-366. Bestellungen nimmt der Verlag oder jede Buchhandlung entgegen.

Verkaufspreis: Das Einzelheft kostet DM 16,-

Produktion: Technik: Klaus Buck (Ltg./180), Wolfgang Meyer (Stellv./887); Herstellung: Otto Albrecht (Ltg./917)

Druck: SOV Graphische Betriebe, Laubanger 23, 8600 Bamberg

Sonderdruck-Dienst: Alle in dieser Ausgabe erschienenen Beiträge sind in Form von Sonderdrucken zu erhalten. Anfragen an Reinhard Jarczok, Tel. 089/4613-185, Fax 4613-774.

Vorstand: Otmar Weber (Vors.), Bernd Balzer

Direktor Zeitschriften: Michael M. Pauly

Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen: Markt&Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon 089/4613-0, Telex 522052, Telefax 089/4613-100

ISSN 0931-8933

Telefon-Durchwahl im Verlag:
Wählen Sie direkt: Per Durchwahl erreichen Sie alle Abteilungen direkt. Sie wählen 089/4613 und dann die Nummer, die in den Klammern hinter dem jeweiligen Namen angegeben ist.

Copyright-Erklärung



Name: ....

....

Anschrift: ....

Datum: ....

Computertyp: ....

Benötigte Erweiterung/Peripherie: ....

Datenträger: Kassette/Diskette ....

Programmart: ....

Ich habe das 18. Lebensjahr bereits vollendet

...., den ....

....

(Unterschrift)

Wir geben diese Erklärung für unser minderjähriges Kind als dessen gesetzliche Vertreter ab.

...., den ....

....

Bankverbindung:

Bank/Postgiroamt: ....

Bankleitzahl: ....

Konto-Nummer: ....

Inhaber des Kontos: ....

Das Programm/die Bauanleitung: ....

das/die ich der Redaktion der Zeitschrift 64'er übersandt habe, habe ich selbst erarbeitet und nicht, auch nicht teilweise, anderen Veröffentlichungen entnommen. Das Programm/die Bauanleitung ist daher frei von Rechten anderer und liegt zur Zeit keinem anderen Verlag zur Veröffentlichung vor. Ich bin damit einverstanden, daß die Markt & Technik Verlag AG das Programm/die Bauanleitung in ihren Zeitschriften oder ihren herausgegebenen Büchern abdruckt und das Programm/die Bauanleitung vervielfältigt, wie beispielsweise durch Herstellung von Disketten, auf denen das Programm gespeichert ist, oder daß sie Geräte und Bauelemente nach der Bauanleitung herstellen läßt und vertreibt bzw. durch Dritte vertreiben läßt.

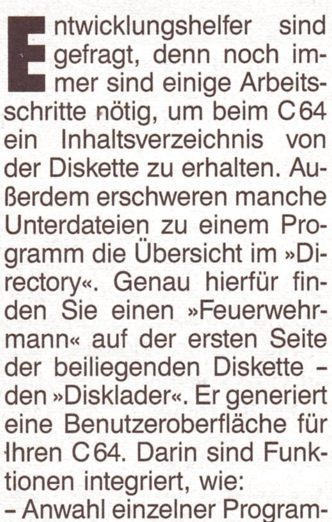
Ich erhalte, wenn die Markt & Technik Verlag AG das Programm/die Bauanleitung druckt oder sonst verwertet, ein Pauschalhonorar.

Disklader – Programme laden mit Komfort

Diskettenoberfläche de Luxe

Keine umständlichen Ladeanweisungen und ein übersichtliches Inhaltsverzeichnis der Diskette auf dem Bildschirm. Unser »Disklader« erfüllt auch gehobene Ansprüche.

von Herbert Großer



Entwicklungshelfer sind gefragt, denn noch immer sind einige Arbeitsschritte nötig, um beim C64 ein Inhaltsverzeichnis von der Diskette zu erhalten. Außerdem erschweren manche Unterdateien zu einem Programm die Übersicht im »Directory«. Genau hierfür finden Sie einen »Feuerwehrmann« auf der ersten Seite der beiliegenden Diskette – den »Disklader«. Er generiert eine Benutzeroberfläche für Ihren C64. Darin sind Funktionen integriert, wie:
- Anwahl einzelner Programme (mit jeweiliger Kurzbeschreibung),
- automatisches Laden und Starten von Diskette oder
- Erkennung der richtigen Diskette bzw. Diskettenseite.

Da sich der Disklader an erster Stelle auf der Diskette zum Sonderheft befindet, genügt es, zum Laden einzugeben:

```
LOAD":*",8
```

Nach der Bestätigung mit <RETURN> dauert es ca. 15 s, bis die Datei im Speicher ist. Sie starten mit RUN und <RETURN>. Anschließend wird das File entpackt (ca.2 s) und es erscheint die Benutzeroberfläche des »Disklader« (s. Abbildung). In der rechten unteren Bildschirmhälfte sehen Sie weiß umrandet den Namen des ausgewählten Programms. Die unterste Bildschirmzeile ist die dazugehörige Kurzerklärung. Zusätzlich finden Sie in der rechten unteren Bildschirmhälfte den Text »Seite 1 auf Disk« oder »Seite 2 auf Disk«. Da Sie die Inhaltsverzeichnisse beider Seiten (ohne die Disk zu wenden) durchblättern können, finden Sie hier den Hinweis, auf welcher Diskettenseite sich das gewählte Programm befindet.



angewähltes programm — Kurzerklärung für das angewählte Programm — Seite auf der Diskette und Feld für Fehlermeldungen

Durch Tastendruck <CRSR aufwärts> bzw. <CRSR abwärts> wählen Sie das nächste oder vorherige Programm. Sie blättern quasi durch den Inhalt der Programme. <HOME> bringt Sie zum ersten Eintrag des Inhaltsverzeichnisses. Selbstverständlich sind nur die Programme verzeichnet, die sich eigenständig laden oder starten lassen.

<RETURN> führt Sie in den Ladeteil. Ist kein Diskettenfehler aufgetreten, erscheint kurzzeitig »00,OK, 00,00« am Bildschirm. Eventuelle Fehleranzeigen bleiben sichtbar am Bildschirm (z.B. »21,READ ERROR, 18,00«= Drive not ready). Sie lassen sich durch einen beliebigen Tastendruck wieder löschen. Schlagen Sie bitte vorher im Handbuch Ihrer Floppy nach und beseitigen Sie den Fehler. Eine andere Art der Fehlermeldung wird durch einen blinkenden Text dargestellt (z.B. »Bitte Disk wenden« oder »Falsche Diskette«). Sind Fehler ausgeblieben, lädt der Disklader das von Ihnen gewählte Programm von der Diskette und startet es. Ladefehler, die in dieser Phase auftreten, werden nicht mehr berücksichtigt: Der Disklader wird vom neuen Programm einfach überschrieben. Sonst könnten wir nur Programme veröffentlichen, die mit der Benutzeroberfläche zusammenarbeiten. Bei vielen Spielen, Tricks oder Tools ist dies aber nicht der Fall.

Für Sie bedeutet dies, nach jedem Starten eines Programms den »Disklader« erneut zu laden. Wer die Benutzeroberfläche verlassen will, gibt <RUN/STOP> ein. Sie befinden sich dann im normalen »Basic« des C64. Für einen Neustart befehlen Sie

```
SYS 12032
```

und bestätigen mit <RETURN>. Dieser Neustart funktioniert auch nach einem Reset, d.h. wenn Sie durch den entsprechenden Taster einen Hardware-Reset ausgelöst haben. Allerdings sollten Sie zwischenzeitlich kein Programm geladen haben, da dies den verwendeten Speicherbereich überschreiben könnte. Laden Sie in diesem Falle den Disklader neu.

Wir haben bei der Programmierung größten Wert auf Kompatibilität mit den unterschiedlichsten Betriebssystemerweiterungen gelegt. Lediglich bei der Gerätekonfiguration C128 mit RAM-Erweiterung und zweiter Diskettenstation sollten Sie die externe Floppy ausschalten. (gr)

Kurzinfo: Disklader

Programmart: Hilfsprogramm zum Laden der Programme auf der beiliegenden Diskette
Laden: LOAD":*",8
Starten: nach dem Laden mit RUN
Steuerung: Tastatur
Programmautor: H. Großer

ob diese Datei kopiert werden soll (<Y>) oder nicht (<N>). Falls Sie die restlichen Files auf der Diskette nicht mehr interessieren, kann man den Suchvorgang mit der Taste <↑> (Pfeil nach oben) abbrechen. Ausgewählte Files werden mit einem Pfeil markiert. Die Dateien können einzeln (separately) oder in der Reihenfolge der Auswahl kopiert werden. Mit <Y> läßt sich zusätzlich die VERIFY-Funktion einschalten. Ohne »Verify« kopiert sich's aber schneller. Das Programm liest die markierten Files in den Speicher (loading) und fordert Sie anschließend auf, die Zieldiskette ins Laufwerk zu legen (Abb. 2). Achten Sie darauf, daß noch genügend Blocks darauf frei sind! Nach Betätigung von <F7> werden alle Files kopiert (saving).

<S> Scratch Files: Die Bedienung dieses Menüpunkts entspricht dem vorhergehenden. Anschließend werden die gewählten Dateien von der Diskette entfernt. Der Name des zu löschenden Files erscheint zur Kontrolle.

<D> Directory: bringt das Inhaltsverzeichnis der aktuellen Diskette auf den Bildschirm.

<O> Order Disk: Alle bekannten DOS-Anweisungen kann man hier in Kurzform ausführen (ohne OPEN- und CLOSE-Befehle). Achtung: Das Programm funktioniert nicht mit dem C 128 D und der Floppy 1571! (Burkhard Graves/bl)

Flucht mit Ausweg

Beim Programmieren geschieht es oft, daß man anstelle einer Cursorbewegung reverse Grafikzeichen erhält: Der sog. Quote-Modus wurde mit dem Setzen von Anführungszeichen aktiviert. Der einzige Weg zurück zur normalen Cursor-Kontrolle führt nur über die RETURN-Taste. Dieses Utility befähigt den Anwender, in der Eingabezeile zu bleiben – und trotzdem den Quote-Modus eingeschaltet zu lassen. Das Programm muß absolut geladen werden:

```
LOAD "ESCAPE-TASTE",8,1
```

Geben Sie anschließend »NEW« ein und aktivieren Sie die Programmierhilfe mit »SYS 49152«. Falls Sie eine Basic-Zeile mit »Gänsefüßchen« einleiten, erscheint in der rechten oberen Bildschirmecke ein »Q« (Quote-Modus aktiv). Es verschwindet, wenn Sie <F7> drücken. Cursor-Bewegungen lassen sich jetzt problemlos innerhalb dieser Zeile ausführen. <RUN/STOP RESTORE> schaltet das Utility wieder ab.

(Christian Spörri/bl)

681 Blocks free

Manchmal möchte man noch ein Programm auf einer relativ vollen Diskette unterbringen, weil es thematisch dazu paßt. Unglücklicherweise ist es z.B. fünf oder sechs Datenblöcke länger als die Diskette noch zuläßt. Bevor Sie jedoch zähneknirschend eine neue Diskette formatieren, laden Sie zuerst unser Utility:

```
LOAD "DISK-FUELLER",8
```

Gestartet wird es mit RUN. Normalerweise ist der verfügbare Speicherplatz von Spur 18 (fürs Directory) noch nicht ganz ausgereizt, auch wenn sich viele Einträge darin befinden. Das Programm schafft Platz auf der Diskette, indem es belegte Blöcke eines Disketten-Files in die Directory-Spur kopiert und die Verbindungszeiger (Linkpointer) entsprechend anpaßt. Wie viele darin noch frei sind, wird Ihnen beim Programmstart mitgeteilt. Im günstigsten Fall können es bis zu 17 Blöcke sein, die Sie dadurch gewinnnen. Nach der Bestätigung mit <J> beginnt die Floppy, die Blocks zu tauschen. Der Bildschirm zeigt Ihnen, welche es sind. Achtung: Verwenden Sie lediglich solche Disketten, deren BAM in Ordnung ist (ggf. vorher ein VALIDATE durchführen!). Ebenso kommt es zu Problemen, wenn die Spur 18 bereits manipuliert wurde: Kommentare oder Abgrenzungsstriche (wie beispielsweise auf den Disketten zu unseren 64'er-Sonderheften) wirken sich in der Zusammenarbeit mit dem »Disk-Füller« verheerend aus. Solche Directory-Zusätze müssen unbedingt vorher geSCRATCHt werden! (Uwe Gerlach/bl)

Ein Luxus-INPUT

Nicht zuletzt an der Gestaltung der Eingaberoutine erkennt man gute Dateiverwaltungsprogramme. Fehler bei der Tastaturbenutzung sollten so weit wie möglich abgefangen werden. Wer die normale INPUT-Routine des Basic 2.0 kennt, weiß, wie wenig Staat man damit machen kann: Die Eingabezeile läßt sich ohne weiteres mit den Cursor-Tasten verlassen, bestimmte Tastaturzeichen sind tabu und erzeugen nach <RETURN> eine Fehlermeldung. Wenn Sie unsere Eingaberoutine in eigenen Programmen verwenden, ist Schluß mit dem Frust.

Laden Sie das Programmbeispiel mit:

```
LOAD "NEUES INPUT",8
```

Man muß lediglich die beiden Variablen IN$ und LT (s. Variablenliste) mit individuellen Werten ausstatten. Dies hat den Vorteil, daß man Standardantworten definieren kann. Nach dem Start mit RUN wartet die Routine auf die Eingabe eines Zeichens. Ein blinkender Cursor wird dadurch simuliert, daß man ein Leerzeichen (CHR$(32)) abwechselnd normal und revers anzeigt. Das Programm prüft alle eingegebenen Zeichen, ob sie erlaubt sind:

- bei <RETURN> wird die Eingabe beendet,
- mit <DEL> reduziert sich die Zeichenkette um ein Byte,
- <SHIFT RETURN> löscht die Eingabezeile und positioniert den Cursor an den Anfang,
- Cursor-Bewegungen und andere Steuerzeichen (<CLR/HOME>) werden ignoriert.

(Willi Burmeister/bl)

Variable	Bedeutung
IN$	Übergabevariable für den einzulesenden String
LT	maximale Länge des Eingabestrings
ZT und ZC	Hilfsvariablen zur Cursor-Simulation
ZL	aktuelle Länge des einzulesenden Strings
Z$	letztes eingegebenes Zeichen
Z	ASCII-Code des gelesenen Zeichens und Laufvariable

Variablenliste zur Routine »Neues Input«

Kurz und bündig

Hires-Grafiken, erzeugt mit herkömmlichen Zeichenprogrammen (Hi-Eddi, Starpainter usw.), verbrauchen 33 Blocks auf der Diskette. Mit diesem Utility kann man sie auf mehr als die Hälfte komprimieren: Byte-Sequenzen, die sich in der Bitmap häufig wiederholen, werden durch kürzere ersetzt.

Laden Sie das Utility mit:

```
LOAD "BMC.EXE",8,1
```

Geben Sie jetzt »NEW« ein. Folgende Funktionen werden durch SYS-Anweisungen (am besten innerhalb eines Programms) aktiviert:

SYS 52736, Filenummer, Geräteadresse, Sekundäradresse, "Bildname,P,W" (speichert die komprimierte Bitmap), Beispiel:

```
SYS 52736,1,8,3,"PIC.1,P,W"
```

SYS 52798, Filenummer, Geräteadresse, Sekundäradresse, "Bildname" (lädt ein kompaktes Bild), Beispiel:

```
SYS 52798,3,8,4,"PIC.1"
```

Vor allem für Grafik-Adventures ist dieses Utility gedacht: Hires-Grafiken können so bis auf 15 Blocks abgespeckt werden! (Hans Haberl/bl)



[3] Zur Vergabe der Ladeattribute muß man die Tasten <F1>, <F3> und <F5> verwenden

Mini-Disklader

Die Benutzeroberfläche, die wir unseren Disketten zu den 64'er-Sonderheften mitgeben, der »Disklader«, ist eine komfortable Hilfe zum Laden von Programmen. Allerdings gilt er jeweils nur fürs aktuelle Heft und ließe sich nur mit entsprechendem Insider-Wissen so verändern, daß er für andere Disketten eingesetzt werden kann. »Menue-Maker« verfolgt das Prinzip des Diskladers und hat den Vorteil, daß Sie **jede** gewünschte Diskette mit einem Lademenü ausstatten können. Ebenso kann man vermerken, ob das Programm mit dem Zusatz »,8« oder »,8,1« geladen werden muß.

Laden Sie das Disketten-Utility mit:

```
LOAD "MENUE-MAKER",8
```

und starten Sie es mit RUN. Das Hauptmenü erscheint. Unter Menüpunkt 3 kann man den Infobildschirm mit Hinweisen zum Programm einschalten. Sie werden aufgefordert, den zweiten Programmpunkt zu wählen: **Übernehmen**. Die einzelnen Directory-Einträge werden angezeigt. Mit folgenden Tasten läßt sich das Inhaltsverzeichnis bearbeiten:

<F1>: Zusatz »,8« (Basic-Programm),
<F3>: Maschinensprache-Files mit »,8,1«
<F5>: Eintrag nicht übernehmen (z.B. Programmteile, die vom Hauptprogramm nachgeladen werden).

Die Bestätigung Ihrer Eingabe erscheint hinter dem Filenamen auf dem Bildschirm (Abb. 3). Im Hauptmenü kann man die eingestellte Konfiguration unter Punkt 4 »Menü speichern« auf Diskette übertragen. Zusätzlich generiert der »Menue-Maker« die Datei »MSD«. Falls Ihnen bei der Vergabe der Ladeattribute ein Fehler unterlaufen ist, wählen Sie erneut Punkt 2. Mit Menüpunkt 1 »Directory einlesen« läßt sich das Inhaltsverzeichnis auf den Bildschirm bringen, Menüpunkt 5 beendet das Programm mit einem Reset. Künftig muß auf dieser Diskette lediglich das Programm »MENU« in den Computer geholt werden. Die Programme lassen sich dann in einem Fenster auswählen. (S. Brüllsauer/bl)

Speicherautomatik

Stromausfall beim Programmieren oder Abtippen eines Listings ist eine mittlere Katastrophe! Vor allem, wenn man aus Bequemlichkeit vergessen hat, in regelmäßigen Intervallen die bisherige Arbeit auf Diskette zu sichern. Bevor Sie sich die Haare raufen: Verwenden Sie lieber unser Utility.

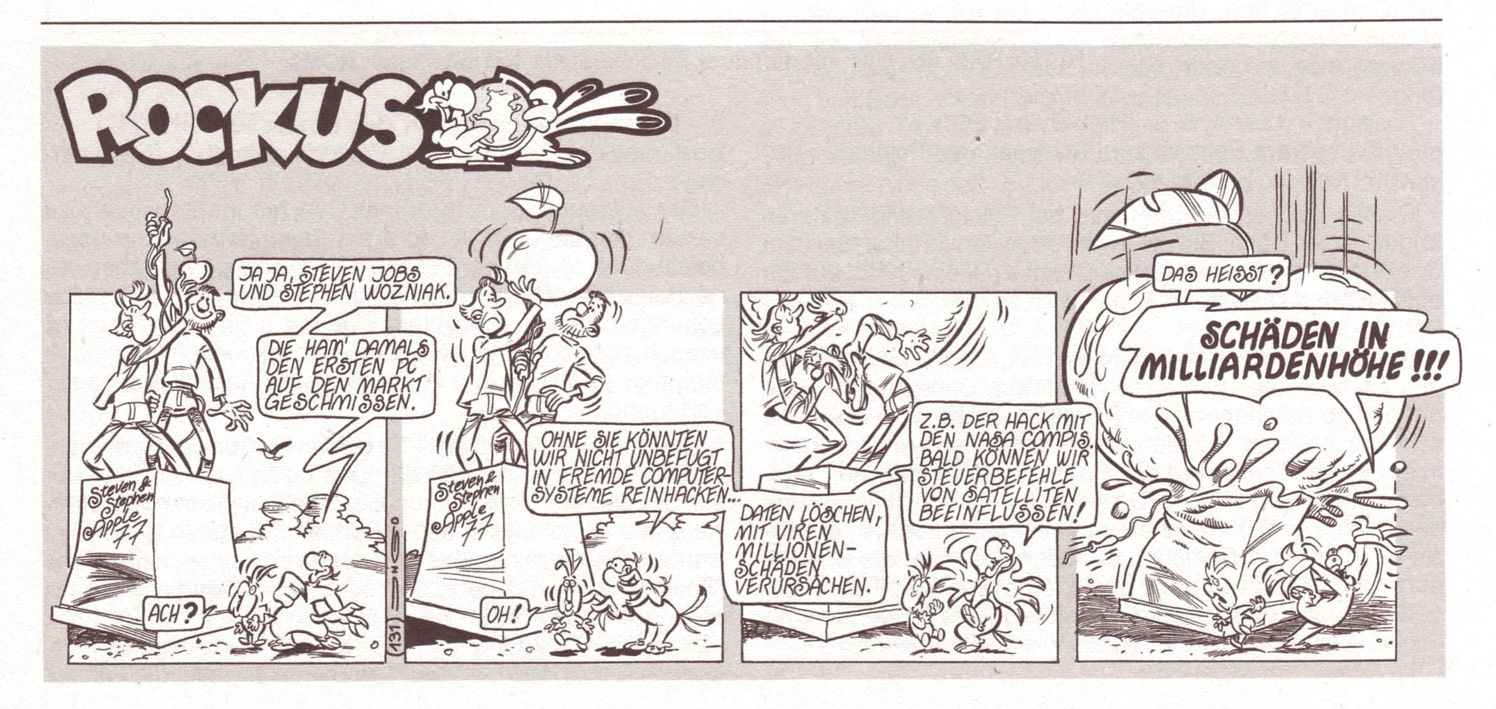
Laden Sie das Programm mit: `LOAD "AUTO-SAVE",8` und starten Sie es mit RUN. Der Maschinenspracheteil »SAVE-OBJ" wird nachgeladen, anschließend müssen Sie den Namen des Programms eingeben, an dem Sie arbeiten möchten. Bei der Frage nach den Speicherintervallen können Sie angeben, nach wie vielen Bytes die Speicherung einsetzen soll (angegebene Zahl x 256). Zwischendurch kann man mit der Taste <@> jederzeit den Speichervorgang auslösen. (Manfred Lins/bl)

Stoppt Programme auf Tastendruck

Die meisten kommerziellen Spiele besitzen eine Pausentaste. Ebenso läßt sich das Scrollen eines Listings über den Bildschirm anhalten, um zu überprüfen, wo der Fehler liegt. Mit »Pause« ist es egal, an welchen Stellen ein Programm angehalten werden soll. Laden Sie das Utility mit:

```
LOAD "PAUSE",8
```

Nach dem Start mit RUN läßt es sich an einen beliebigen Speicherbereich im RAM verschieben (voreingestellt ist »49152«). Mit <F1> wird der »Zwischenstopp« während des Ablaufs eines Programms oder Listings aktiviert, jede andere Taste hebt die Sperre wieder auf. (Georg Kramer/bl)



Register, Pointer und Vektoren

Comput

Die oft zitierte Zero-Page (Null-Seite) ist vollgepropft mit den wichtigsten Grunddaten. Erst diese, zusammen mit den Maschinenprogrammen des Basic-Interpreters und des Betriebssystems im ROM, erwecken Ihren C64 zum Leben. Die meisten Speicherstellen sind für Basic und das Betriebssystem reserviert. Ohne die Daten der Zero-Page würde nach dem Einschalten kein einziges Zeichen am Bildschirm erscheinen. Bei unserer ausgiebigen Wanderung durch die ersten 256 Byte des Computers werden Sie immer wieder auf drei Bezeichnungen stoßen:

1. Pointer (Zeiger auf den Beginn von Daten)
2. Vektoren (Zeiger auf Maschinenprogramme)
3. Flags (abgelegter Zahlenwert, um auf eine Rechenoperation zu reagieren, bzw. sich das Resultat zu merken)

Pointer und Vektoren stehen jeweils in benachbarten Speicherstellen. Da pro Speicherstelle nur 256 Werte zur Verfügung stehen, der Speicher aber 65535 »Hausnummern« hat, sind zwei Speicherstellen (Low-/High-Byte) nötig, um höhere Werte anzupeilen. Die Zellen werden dabei zu einer Zahl verknüpft. Sehen wir uns das am Beispiel der Speicherstellen 43 und 44 an. Beide zusammen ergeben einen Pointer auf den Basic-Anfang, also auf die Speicherposition, an der ein Basic-Programm geladen und gestartet wird. Die niedrigere von beiden ist (egal welchen Wert die hat) die Speicherstelle 43 und wird Low-Byte genannt. Ihr Inhalt wird unverändert übernommen und deckt damit den Bereich »0« bis »255«. Die zweite (44) wird High-Byte genannt und der Inhalt mit 256 multipliziert. Das heißt, steht in dieser Speicherstelle »1«, bedeutet dies für den Mikroprozessor »256«, »2« ergibt »512« und, um den letztmöglichen Betrag zu verwenden, »255« ergibt »65280«. Addiert man den Inhalt von 43 dazu, läßt sich alles zwischen »0« und »65535« darstellen und damit der gesamte Speicher des C64 erreichen. Um solche Pointer zu berechnen, braucht man folgende Formel:

Position = Low-Byte + (High-Byte x 256)

oder in unserem Beispiel zum Auslesen des Pointer:

```
PRINT PEEK(43)+PEEK(44)*256
```

Dieses Rechenverfahren wird bei Pointern und Vektoren angewendet. Damit Sie die Werte auch komfortabel aus dem Speicher lesen können, befindet sich ein kleines Tool auf der beiliegenden Diskette. Geladen wird es mit

```
LOAD"Z-PAGE",8
```

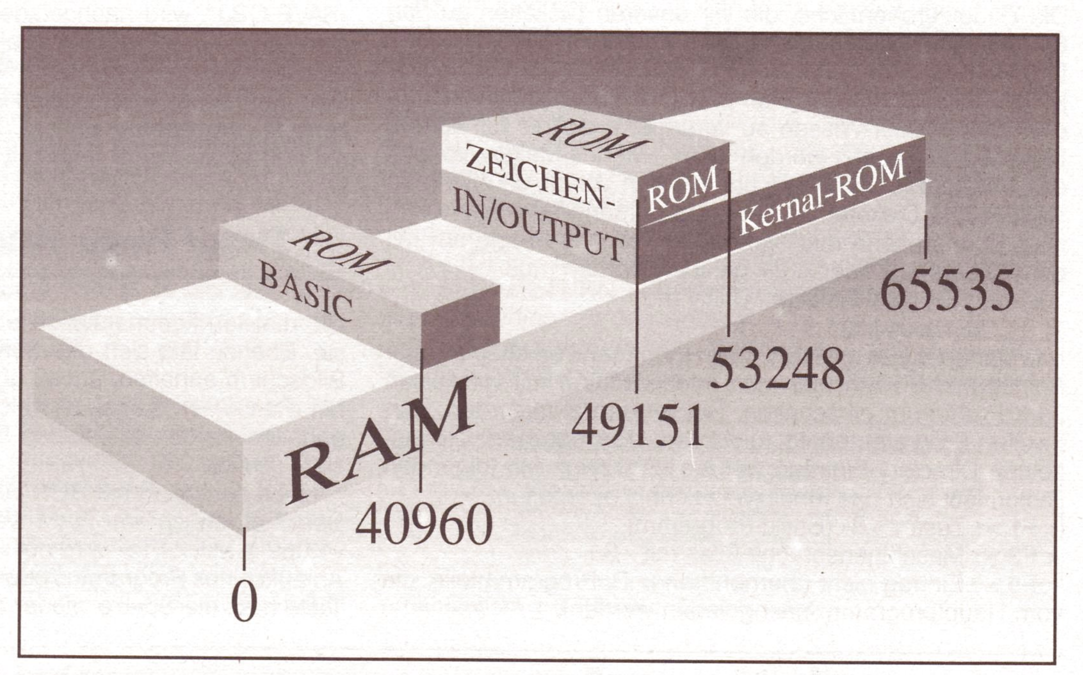
Nach der Eingabe von RUN wird zunächst ca. drei Sekunden ein Maschinenprogramm installiert. Danach geben Sie nur ein, ab welcher Position Sie die Ausgabe wünschen und schon erscheinen 15 Zeilen mit je vier Spalten am Bildschirm. In der ersten Spalte wird die Nummer der Adresse ausgegeben, dann folgen Inhalt in Dezimal (#), Hexadezimal ($) und das Bitmuster (%). Da es aber manchmal sinnlos ist, ein Basic-Programm im Speicher zu haben, läßt sich die Maschinenroutine quasi von Hand bedienen. Wenn Sie NEW eingegeben haben, oder eines der anderen Tools verwenden, erreichen Sie die gleiche Ausgabe wie in Basic durch:

```
POKE 900,Start:SYS49152
```

Beachten Sie dabei, daß für »Start« die Speicherposition eingegeben werden muß, mit der Sie beginnen möchten.

Bei der Beschreibung der Zero-Page steht in der fettgedruckten Zeile jeweils die Speicherposition, eine Kurzbezeichnung und wenn sinnvoll, in Klammern der Normalwert. Bei einigen Speicherstellen fehlen die Angaben des Normalwerts. Diese Adressen haben keine definierten Werte oder ändern sich ständig.

In den nächsten Zeilen folgen eine Kurzbeschreibung und, wenn von Basic aus möglich, einzelne Tricks. Auf eine ausführliche Beschreibung der Band-(Kassetten-)Routinen wurde aus Platzgründen verzichtet.



[1] Das RAM des C64 mit den über Speicherstelle 1 erreichbaren ROMs

0 – Datenrichtungsregister des Prozessor-Ports (47)

Bestimmt die Richtung des Prozessor-Ports (= Speicherstelle 1).

Der Mikroprozessor (6510) im C64 hat im Gegensatz zu seinem Vorgänger 6502 sechs programmierbare Ein-Ausgabeleitungen eingebaut. Er kann damit wichtige Aufgaben ohne zusätzliche Bausteine erledigen. Warum aber nur sechs Leitungen und nicht acht (ein Byte hat 8 Bit)? Die Frage ist einfach zu beantworten. Der Chip selbst könnte 8 Bit verkraften, aber es stehen ihm nur noch sechs Anschlußbeine zur Verfügung.

Die Speicherstelle 0 stellt für diesen Port das entsprechende Datenrichtungsregister dar. Das heißt, jede einzelne der sechs Leitungen kann durch Beschreiben dieser Speicherstelle als »Empfänger« oder »Sender« für Daten geschaltet werden. Die einzelnen Bits sind denen des eigentlichen Ports (Speicherstelle 1) zugeordnet. Benutzt werden die Bits 0 bis 5. Es gilt für alle 6 Bit:

- Bit auf 0 = Eingang
- Bit auf 1 = Ausgang

Beim Einschalten schreibt das Betriebssystem die Zahl 47

erstart

Die ersten 256 Byte im Speicher des C64 sind voller Rätsel. Schreibt man einen Wert hinein, macht der Computer plötzlich seltsame Dinge. Wir haben die Erklärung dafür.

in dieses Register. Als Dualzahl ausgeschrieben wird jeder einzelne Zustand (Bit) beschrieben. Die Wertigkeit verläuft hier von rechts nach links. Das niedrigstwertige Bit (ganz rechts) wird mit Bit 0 bezeichnet.

Dezimal 47 = Dual 00101111

Wir sehen, nur Bit 4 ist als Eingang geschaltet. Bit 6 und 7 interessieren nicht, sie sind wie oben beschrieben ohne Funktion.

1 - Input-Output (55)
Ist der Prozessor-Port des Mikroprozessors. Seine sechs Ein-Ausgabeleitungen sind mechanisch herausgeführt und lassen sich abfragen oder beschreiben, je nachdem wie ihre Datenrichtung in Register 0 festgelegt ist. Im C64 sind diese Leitungen festen Funktionen zugeordnet:

Bit	Funktion
0	Basic-ROM =1, RAM = 0
1	Kernal-ROM =1, RAM = 0
2	I/O = 1, Zeichensatz = 0
3	Datenausgabe von Datasette
4	Taste von Datasette gedrückt = 0 nicht gedrückt = 1
5	Motor an = 1, Motor aus = 0
6	unbenutzt = 0
7	unbenutzt = 0

Nach dem Einschalt-Reset ist die Beschaltung dezimal 55 = dual 00110111. Das heißt Basic- (40960 bis 49151) und Kernel-Bereich (57344 bis 65535) sind auf ROM geschaltet (Abb. 1). Zum Verständnis dieser Konfiguration muß man wissen, daß im C64 für alle genannten Bereiche nicht nur ROM (Nur-Lese-Speicher), sondern auch RAM, also Schreib-Lese-Speicher, zur Verfügung steht. Ein RAM verliert im Gegensatz zum ROM beim Aus- und Wiedereinschalten seine Daten. Ohne diese Daten könnte aber der Mikroprozessor nicht arbeiten.

Im Speicherbereich 53248 bis 57343 (Bit 2) wird das Input-Output-ROM erreicht. Die anderen Daten werden nur bei Kassettenoperationen genutzt.

Vorsicht, wenn Sie diese Speicherstelle mit

```
POKE1,53
```

auf RAM umschalten, verirrt sich der Mikroprozessor im Labyrinth seines Speichers und ist auf normalem Weg zu keiner Reaktion mehr zu bewegen (53 = 00110101). Nur Aus- und Einschalten (oder ein Reset mittels Taster) bringt ihn wieder auf Trab. Aber es gibt eine Lösung: Laden Sie doch mal

```
LOAD"ROM TO RAM",8,1
```

und starten Sie mit SYS 49152. Danach geben Sie NEW ein, um die Pointer wieder geradezusetzen. Wenn Sie jetzt mit obigem POKE umschalten, passiert scheinbar überhaupt nichts. Der Grund ist, unser gerade eben geladenes Maschinenspracheprogramm hat das ROM ins RAM kopiert. Danach steht in beiden Speicherbereichen die gleiche Information und der Computer stürzt nicht ab. Den letzten POKE hätten Sie sich übrigens sparen können, er ist schon in der Routine eingebaut. Praktischen Nutzen hat das kleine Tool für Änderungen am Betriebssystem. Das Kopieren in Basic würde zwar auch funktionieren, dauert aber extrem lange. Probieren Sie es doch mal und geben Sie ein:

```
10 FORI=40960TO49151:POKE I,PEEK(I):NEXT
20 FORI=57344TO65535:POKE I,PEEK(I):NEXT
```

Starten Sie danach mit RUN. Sie werden sich vielleicht über die einfache Art des Kopierens in der FOR-NEXT-Schleife wundern. Aber wir nützen nur eine Eigenheit unseres Computers aus. Er erkennt von selbst, daß er ein ROM nicht mit POKE beschreiben kann - und mit PEEK liest er aus dem RAM, da die Speicherstelle 1 den Wert 55 eingetragen hat.

Inzwischen dürfte Ihr C64 auch mit seiner POKErei fertig geworden sein. Wundern Sie sich nicht, die Routine braucht über eine Minute.

2 - unbenutzt (0)

3/4 - Vektor für Umwandlung von Fließkomma nach Fest (170/177 = 45482)
Wird für interne Rechenumwandlungen verwendet. Zeigt auf eine Routine im Betriebssystem, die in Basic ständig bei Zahleneingaben und Umrechnungen verwendet wird. Daher läßt sie sich nur aus Maschinenprogrammen nützen.

5/6 - Vektor für Umwandlung Fest nach Fließkomma (145/179 = 45969)
Wird für interne Rechenumwandlungen verwendet. Ist die Umkehrung der Funktion des Vektors 3/4. Auch diese Funktion läßt sich nur aus Maschinenprogrammen verwenden.

7 - Suchzeiger
Wird in den Basic-Routinen als Zwischenspeicher zur Prüfung von Texteingaben verwendet. Normalerweise wird der ASCII-Wert der geprüften Zeichen abgelegt. Da aber auch READ, AND, INT, EXP u.a. diese Zelle mitbenutzen, ist sie aus Basic nicht zu verwerten.

8 - Suchzeiger für Hochkomma-Flag
Ähnlich der Speicherstelle 7 dient diese Speicherstelle als Zwischenspeicher. Sie wird hauptsächlich bei der Umwandlung von Basic-Befehlen in Tokens (Befehlscode) verwendet.

9 - Spalte TAB
Wird von den Befehlen TAB und SPC benutzt und zeigt die Spaltenposition des Cursors vor der Ausführung der Befehle.

10 - Load/Verify
In dieser Zelle steht 0, wenn geladen wird und 1 bei Verify. Da die Routinen für LOAD und VERIFY identisch sind, unterscheidet das Betriebssystem anhand dieser Speicherstellen zwischen beiden Modi (vergl. auch 147)

11 - Flag für Anzahl DIMs und für Eingabe
Wird zur Berechnung für die Anzahl der Felder und nach Abschluß einer Tastatureingabe als Zwischenspeicher verwendet.

12 - Flag für DIM
Zwischenspeicher für Basic-Routinen zur Unterscheidung zwischen Variable oder Feld. Außerdem Kennung, ob Feld bereits dimensioniert ist oder als undimensioniertes Feld bereits elf Elemente hat.

13 - Typflag Numerisch oder String
Gibt dem Basic-Interpreter die Art der zu verarbeitenden Daten an.

14 – Typ Integer oder Real
Arbeitet mit Zelle 13 zusammen. Für ganze Zahl steht 128, für eine Gleitkommazahl 0.
15 – Hochkomma-Flag bei LIST
Schaltet bei LIST zwischen Token-Umwandlung in Text und Textausgabe (nach Gänsefüßchen) um. Nach der Garbage-Collection (Entfernung der nicht mehr belegten Variablen) erscheint eine Markierung. Ist danach ein Speichern neuer Variablen nicht möglich, wird OUT OF MEMORY angezeigt.
16 – Flag für FN
Ist für den Basic-Interpreter beim Suchen nach Variablen das Unterscheidungs-Flag zwischen normalen Variablen und selbstdefinierten Funktionen.
17 – INPUT/GET/READ
Da Teile der drei Routinen identisch sind, unterscheidet das Betriebssystem durch diese Zelle, in welche unterschiedlichen Programmabschnitte verzweigt werden muß:
0 = INPUT
64 = GET
152 = READ
18 – Vorzeichen TAN/SIN/COS
Da die Vorzeichen bei trigonometrischen Funktionen in den einzelnen Kreisquadranten unterschiedlich sind, wird mit dieser Speicherstelle der Wechsel gekennzeichnet. Zur Verdeutlichung dient folgendes Beispiel:

```
10 FOR I=0TO10 STEP 0.01
20 PRINT PEEK(18); INT(I*100)/100;SIN(I):NEXT
```

nach dem Start mit RUN werden Sie feststellen, daß kurz nach dem Wechsel der 1. Zahl von 0 auf 255 sich auch das Vorzeichen der dritten Zahl ändert.
19 – Aktives I-O-Gerät (0)
gibt dem Betriebssystem an, welches Peripheriegerät aktiv ist. Zur Debatte stehen Tastatur, Datasette, RS232, User-Port, Bildschirm, Drucker oder Floppylaufwerk. Außerdem ist das Flag ausschlaggebend für die Anweisung »PRESS PLAY ON TAPE« und das Fragezeichen bei Input. Zur Erklärung dazu befindet sich ein Demoprogramm auf der beiliegenden Diskette. Laden Sie es mit

```
LOAD"ADRESSE 19",8
```

und starten Sie mit RUN. Mit diesem kleinen Programm wird Ihnen gezeigt, wie Sie das lästige Fragezeichen bei Input einfach abschalten.
20/21 – Integer-Adresse
enthält die Zeilennummer bei Bearbeitung der Befehle LIST,GOTO,GOSUB und ON. Da Zeilennummern bis maximal 63999 gehen dürfen, benötigen diese Routinen eine 2-Byte-Darstellung. Nach Bearbeitung dieser Routinen steht hier grundsätzlich der Wert 20.
22 – Pointer Stringstack
zeigt auf den nächsten freien Speicherplatz im vorläufigen String-Zwischenspeicher (Temporary String Descriptor Stack, Zellen 25-33).
23/24 – Pointer letzter String
Zeiger auf die Adresse der letzten Zeichenkette im Temporary String Descriptor Stack.
25-33 – Temporary String Descriptor Stack
Stapelzeiger für Angaben über vorläufige Zeichenketten. Als vorläufig gelten Strings dann, wenn sie noch keiner Variablen zugeordnet sind:

```
PRINT"MAHLZEIT"
```

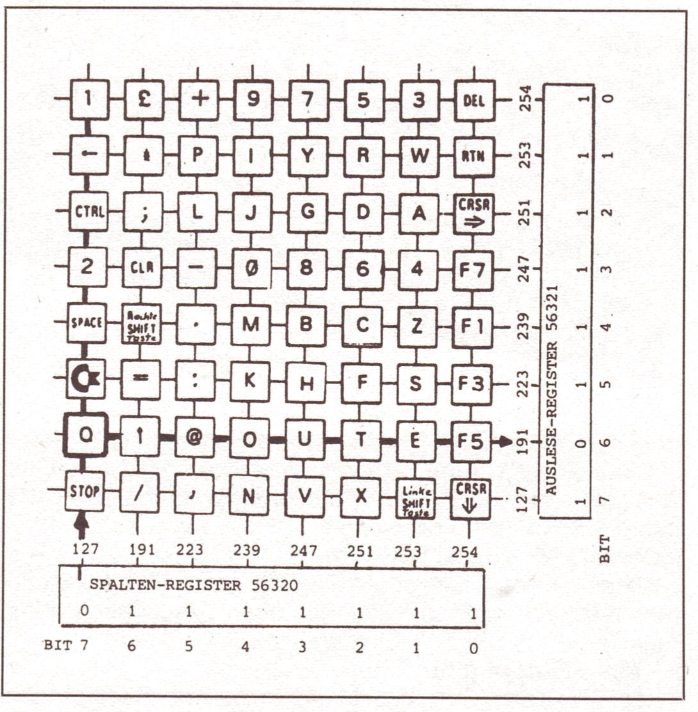
ist ein typisches Beispiel, wogegen

```
A="MAHLZEIT"
```

der Variablen »A« zugeordnet wird und danach irgendwo im Speicher abgelegt wird. In dem Speicherbereich 25-33 wird nicht der String (MAHLZEIT), sondern die Position, wo er bei der Eingabe ist, festgehalten.
34 bis 37 – Pointer div. Zwecke
werden vom Basic-Interpreter für die verschiedensten Zwischenergebnisse und Flaggen genutzt.
38 bis 42 – Akku 1
wird bei Multiplikation und Division als Arbeitsspeicher verwendet.
43/44 – Pointer BASIC-Start (210/32 = 2049)
zeigt auf den Anfang des Basic-Speichers. An diese Position wird ein Programm mit

```
LOAD"xxx",8
```

geladen. Daraus ergeben sich in Basic Anwendungen, wie



[2] Tastenanordnung (Matrix) des C64

Schützen eines Speicherbereiches, z.B. für einen Zeichensatz oder ein Maschinenprogramm. Auch das Zusammenführen mehrerer Programmteile wird möglich (siehe Merge, S. 2).
45/46 – Pointer Variablenstart (3/8 = 2051)
zeigt auf die Anfangsadresse des Speicherbereichs für Variable (keine Felder). Die Variablen beginnen nur durch zwei Nullen getrennt unmittelbar nach einem Basic-Programm. Daher kann man auch die Endadresse des Programms berechnen:

```
PRINT (PEEK (45)+ PEEK (46)*256)-2)
```

Subtrahiert man von diesem Wert die Startadresse,

```
PRINT((PEEK(45)+PEEK(46)*256)-2)-
(PEEK(43)+PEEK(44)*256)
```

erhält man die Länge des Programms im Speicher.
47/48 – Pointer ARRAY-Start (3/8 = 2051)
zeigt auf die Anfangsadresse des Speicherbereichs für Felder (Arrays).
49/50 – Pointer ARRAY-Ende (3/8 = 2051)
zeigt auf die Endadresse (+1) des Speicherbereichs für Felder.
51/52 – Pointer RAM-Ende (0/160 = 40960)
zeigt auf das Ende des frei verfügbaren Variablenspeichers. Also nach einem Neustart oder einem CLR auf 40960.
53/54 – Hilfs-Pointer für Strings
In diesem Zeiger steht die Adresse der zuletzt eingelesenen Variablen.
55/56 – Pointer BASIC-RAM-Ende (0/160 = 40960)
zeigt auf das Ende des in Basic verfügbaren Gesamtspeichers. Auch hier kann durch Herabsetzen des Wertes Platz für Maschinenprogramme gemacht werden. Nötig ist das, da der Basic den Text der Variablen vom oberen Bereich, also von diesem Pointer, abwärts anlegt.

57/58 – BASIC-Zeilennummer
enthält die Zeilennummer der gerade bearbeiteten Basic-Zeile. In Erweiterungen zeigt beim Tracebefehl (Fehler nach Zeilennummer suchen) ein Maschinenprogramm diese Speicherstellen an.
59/60 – Zeilennummer für CONT
enthält die Nummer der letzten ausgeführten Zeile nach END, STOP oder Abbruch durch die Taste <RUN/STOP>.
61/62 – Pointer für CONT
zeigt auf die Speicherposition der letzten ausgeführten Zeile nach END, STOP oder <RUN/STOP>.
63/64 – Zeilennummer für DATA
enthält die Zeilennummer der gerade eingelesenen Data-Zeile.
65/66 – Pointer auf nächstes DATA
zeigt auf die Speicherposition der nächsten Data-Zeile.
67/68 – Herkunft Eingabe
zeigt auf die Adresse, von der sich die Befehle INPUT,GET und READ die Zeichen holen.
69/70 – Name Variable
Name der gerade bearbeiteten Variable
71/72 – Adresse Variable
zeigt auf die Adresse des Inhalts der gerade aufgerufenen Variablen.
73/74 – Pointer Schleifenvariable
enthält bei FOR/NEXT die Adresse der Schleifenvariable, bevor sie im Prozessorstapel verarbeitet wird, und dient außerdem als Zwischenspeicher bei etlichen Basic-Befehlen.
75/76 – Zwischenspeicher Programmzeiger
wird als Zwischenspeicher für Zeiger bei READ und mathematischen Funktionen verwendet.
77 – Maske für Vergleichsoperationen
bei der Auswertung von AND und OR wird hier eine Maske angelegt, die das Ergebnis der logischen Operation angibt. Läßt sich in Basic nicht abfragen.
78/79 – Pointer für FN
zeigt auf die Adresse, ab welcher der Wert einer selbstdefinierten Funktion gespeichert ist.
80-82 – Stringdescriptor
zeigt auf den provisorischen Speicherplatz einer gerade bearbeiteten Zeichenkette.
83 – Flag für Garbage Collection
enthält bei der Speicheraufräumung einen Wert, der angibt, welche Länge (3 oder 7 Byte) die Variable hat.
84 bis 86 – Vektor für Funktionen
in Zelle 84 steht 76, der indirekte Maschinenbefehl JMP (xx), 85 und 86 zeigen auf die Adresse in einer Tabelle. Die Adresse zeigt auf die abzuarbeitende Routine.
87 bis 96 – Akku 3 und 4
Zwischenspeicher für arithmetische Operationen.
97 bis 102 – FAC
Fließkomma-Akkumulator Nr. 1. Spielt eine zentrale Rolle bei der Verarbeitung von Zahlen.
97 = Exponent, 98-101 = Mantisse, 102 = Vorzeichen.

103 – Zähler Polynom
Vorzeichenspeicher bei Umwandlung ASCII in Fließkomma und Zähler bei der Polynomberechnung.
104 – Rundungsbyte FAC
Überlauf aus FAC. Wird nur verwendet, wenn sich nach der internen Berechnung eine darstellbare Zahl ergibt (kleiner 1,70141183 E 38).
105 bis 110 – ARG
Fließkomma-Akkumulator Nr. 2. In ihm steht das Argument bei mathematischen Operationen.
111 – Flag für Vorzeichen FAC/ARG
bei gleichen Vorzeichen steht hier >0<, bei unterschiedlichen >255<.
112 – Rundungsbyte
gehört zu FAC. Überzählige Stellen aus der Mantisse werden hier abgelegt und später wieder geholt, um die Genauigkeit zu erhöhen.
113/114 – Pointer div. Zwecke
wird von vielen Routinen für interne Operationen verwendet.
115/138 – CHRGET
holt nächstes Zeichen eines Basic-Textes. Diese Routine ist das Kernstück des Basic-Interpreters. Beim Einschalten wird sie aus dem ROM kopiert und ist als einziges Maschinenprogramm des Basic-Interpreters in der Lage, sich selbst zu verändern. Durch diese Selbstmodifizierung kann sie den gesamten Speicher erreichen. Dadurch wird es erst möglich, Befehle aus Basic-Zeilen oder per Direkteingabe zu verarbeiten.
139 bis 143 – letzter RND-Wert
enthält beim Einschalten die Werte 128, 79, 19, 82 und 88. Nach Aufruf der RND-(x) Funktion wird das neue Ergebnis als Ausgangswert für die nächste Funktion abgelegt.
144 – Status ST
enthält die Statusvariable, die Sie auch mit PRINT(ST) aufrufen können. Die Bedeutung bei einzelnen gesetzten Bits ist bei Floppy/Drucker:
0 - Fehler beim Schreiben
1 - Fehler beim Lesen
6 - Datenende
7 - Fehler DEVICE NOT PRESENT
145 – Flag für STOP (255)
dient dem Betriebssystem als Zwischenspeicher für Programmunterbrechungen durch <RUN/STOP>. Bei der Tastaturabfrage wird 60mal in der Sekunde das laufende Programm unterbrochen und überprüft, ob eine Taste gedrückt ist:

Alle Tasten sind zunächst geöffnete Schalter. Sie sind raffiniert in einer Matrix (Abb. 2) mit den Registern der CIA verbunden. Drücken Sie nun in Gedanken die Taste <Q> von Abb. 1. Sie verbinden elektrisch Bit 7 des Spaltenregisters mit Bit 6 des Ausleseregisters. Schaltet man nun Bit 7 des Spaltenregisters auf Ausgabe und überprüft der Reihe nach beim Ausleseregister, ob von Bit 0 bis Bit 7 ein Signal (Strom) ankommt, kann die gedrückte Taste identifiziert werden. In der Interrupt-Routine geschieht dies. Und zwar für Bit 7 bis Bit 0 des Ausgaberegisters. Bei jeder Ausgabe wird bei allen Stellen des Ausleseregisters überprüft, ob Strom fließt. Ist dies der Fall, werden beide Werte abgelegt und später in den Buchstaben-Code (ASCII) umgewandelt. Bit 7 kommt dabei eine besondere Aufmerksamkeit zu. Bei ihm wird der Wert des Ausleseregisters grundsätzlich in Speicherstelle 145 abgelegt. Ob eine Taste gedrückt war oder nicht, ist dabei nicht von Bedeutung. Erst später wird überprüft, ob Bit 7 des Ausleseregisters etwas empfangen hat. Ist das der Fall, war die Taste <RUN/STOP> gedrückt und ein laufendes Programm wird unterbrochen.

Den Effekt, daß in Zelle 145 immer der aktuelle Spaltenwert der 1. Spalte steht, können wir uns zunutze machen. Wir können mit PEEK(145) auch andere Tasten erkennen:

254 = <1>	
253 = <Pfeil links>	239 = <SPACE>
251 = <CTRL>	223 = (C= >
247 = <2>	191 = <Q>

Die Taste <RUN/STOP> läßt sich nicht abfragen, da vorher ein Basic-Programm abgebrochen wird. Und ohne Programm läßt sich nichts abfragen, da nach Eingabe des Befehls <RETURN> gedrückt werden muß. Anschließend ist keine Taste mehr gedrückt. Wenn Sie es probieren, erhalten Sie auch prompt »255« - den Wert für keine Taste.
146 – Zeitkonstante für Band
wird nur bei Bandoperationen verwendet.
147 – Flag LOAD/VERIFY
dient zur Unterscheidung zwischen Lade- oder Speicher-

operationen. Ihr Wert ist identisch mit dem in Speicherzelle 10.

148 – Flag für IEC
zeigt dem Betriebssystem, ob ein Zeichen für Drucker oder Floppy im Ausgabepuffer (149) steht.

149 – Puffer IEC
enthält das Zeichen, das in Zelle 148 gemeldet wird.

150 – Flag EOT
Zwischenspeicher nur bei Bandoperationen.

151 – Puffer X-Register bei Lesen Band
wird nur bei Bandoperationen verwendet.

152 – Anzahl offener Files
enthält die Anzahl der mit OPEN geöffneten Dateien (Drucker, Floppy usw.), darf aber in Basic nicht zurückgesetzt werden.
SYS 62255 schließt alle geöffneten Files. Beachten Sie bitte, daß max. 10 Dateien geöffnet sein dürfen.

153 – aktives Eingabegerät
enthält die Gerätenummer für das momentan gültige Eingabegerät:

0 = Tastatur
1 = Datasette
2 = RS232 (über Userport-Kabel)
3 = Bildschirm
4/5 = Drucker
8-11 = Floppylaufwerk(e)

Diese Gerätenummern sind festgelegt und werden bei allen Ein- und Ausgaben verwendet.

154 – aktives Ausgabegerät
wie 153, nur für Ausgabe.

155 – Parität Band
Fehlerkontrolle nur bei Bandoperationen.

156 – Flag für Parität richtig
wird nur bei Bandoperationen verwendet.

157 – Flag Direct/Programm
unterscheidet die Ausgabe der Fehlermeldungen zwischen Betriebssystem und Basic-Interpreter.

158/159 – Pass1/2 Checksumme für Band
wird nur bei Bandoperationen verwendet.

160 bis 162 – Time
enthält den Wert der internen Uhr, der mit PRINT TI als Zahl oder mit PRINT TI$ als sechsstelliger Zeitwert im Format HHMMSS (H = Stunde, M = Minute, S = Sekunde) abgefragt werden kann.

163/164 – Bitzähler seriell/Band
wird bei Ein- und Ausgabeoperationen verwendet

165 – Bitzähler Band schreiben
wird nur bei Bandoperationen verwendet.

166 – Pointer Bandpuffer
nur bei Bandoperationen

168 bis 171 – Puffer I/O (Band/RS232)
bei Eingabe über RS232 und bei Ein- und Ausgabeoperationen von Band werden hier Ergebnisse zwischengespeichert.

172/173 – Pointer Einfügen/Scroll/Bandpuffer
zeigt auf die Anfangsadressen für Ein-Ausgabe, wird als Zwischenspeicher für den Bildschirmeditor eingesetzt und bei Bandoperationen.

174/175 – Pointer Ende speichern und Scroll
Endadresse für Speicheroperationen (siehe Speicherstelle 183) und Zwischenspeicher für den Bildschirmeditor und bei Bandoperationen.

176 bis 177 – Synchronisierung lesen von Band
Wird nur bei Bandoperationen verwendet.

178/179 – Pointer auf Bandpuffer (60/3 = 828)
zeigt auf Beginn des Kassettenpuffers

180 – Bitzähler für Band
Wird nur bei Bandoperationen verwendet.

181 – nächstes Bit RS232
enthält das jeweils nächste zu übertragende Bit

182 – Puffer für Ausgabebyte RS232
enthält das Byte, das bitweise in 182 ausgegeben wird.

183 – Länge Filenamen
Bei LOAD, SAVE und VERIFY steht hier die Länge des aktuellen Filenamens. Diese Speicherstelle läßt sich auch aus Basic nutzen. Dazu befindet sich auf der beiliegenden Diskette ein Demoprogramm. Sie laden es mit

```
LOAD"LADEN",8
```

Bevor Sie starten, beachten Sie: Wenn Sie dieses Programm in eigenen Routinen verwenden, muß die Zeile 5 als erste Zeile im Programm stehen. Nach REM tragen Sie den Namen ein, unter dem Sie speichern wollen. Vergessen Sie das Leerzeichen nach REM nicht!

Absolutes Laden von Files

In Zeile 10 tragen Sie die Werte der Startadresse im Low-/High-Byte-Format in S1/S2 ein. Das gleiche gilt für die Endadresse (+1) in E1/E2. Schließlich gehört noch für »F« in Zeile 30 die Länge des Filenamens eingetragen. Dazu ein Beispiel:
Wir wollen den Bereich 49152 bis 53247 unter dem Namen »TEST« abspeichern. Dann muß unser Programm lauten:

```
5 REM TEST
10 POKE193,0:POKE194,192:POKE174,0:POKE175,208
20 POKE187,PEEK(43)+6:POKE188,PEEK(44)
30 POKE183,4
40 POKE186,8:POKE185,3:SYS62954
```

Um Ihnen auch das Laden an eine beliebige Position des Speichers zu erlauben, laden Sie

```
LOAD"LADEN",8:
```

und starten das Programm mit RUN. Sie werden zuerst nach dem Filenamen gefragt, dann nach der Ladeadresse. Nach beiden Eingaben wird das gewünschte File geladen.

184 – logische Filenummer
Nummer der derzeitigen Datei bei einem OPEN-Befehl.

185 – Sekundärnummer
Derzeitige Sekundärnummer (siehe Speicherstelle 183). Für die Bedeutung:

186 – Gerätenummer
Derzeitige Gerätenummer (siehe Speicherstelle 183). Für die Bedeutung der Werte sehen Sie bitte unter 153 nach.

187/188 – Pointer auf Filename
Zeiger auf derzeitigen Filenamen (siehe Speicherstelle 183).

189 – Prüf-Byte RS232
Zwischenspeicher für Parity-Prüfung und Bandoperationen.

190 bis 192 – Zähler/Puffer seriell/Flag Bandmotor
nur bei Bandoperationen.

193/194 – Startadresse I/O
Anfangsadresse für Ein- bzw. Ausgabeoperationen (siehe Speicherstelle 183).

195/196 – Endadresse I/O
nur für Bandoperationen

197 – gedrückte Taste
Tastencode der zuletzt gedrückten Taste. Wird vom Betriebssystem aus Zelle 203 übernommen. Dient zur Unterscheidung für Tastaturwiederholungen.

198 – Anzahl Tastendrücke (0)
enthält die Anzahl der nicht ausgewerteten Zeichen im Tastaturpuffer (631-640). Wird diese Speicherstelle auf 0 zurückgesetzt, läßt sich sehr einfach auf einen Tastaturdruck warten:

```
POKE198,0:WAIT198,1
```

Mit einem anschließenden GET Z$ läßt sich dieser anschließend auswerten.

199 – Flag für RVS (0)
steht in dieser Zelle nicht Null, werden alle folgenden Zeichen revers ausgegeben. Laden Sie dazu:

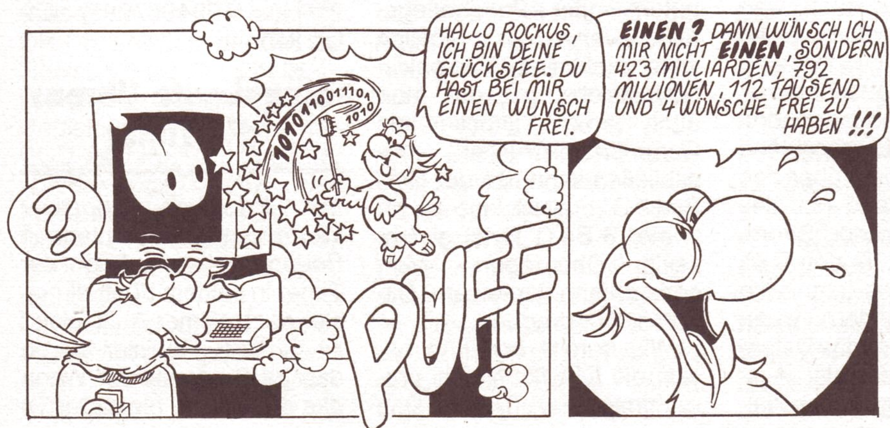
```
LOAD"ADRESSE 199",8
```

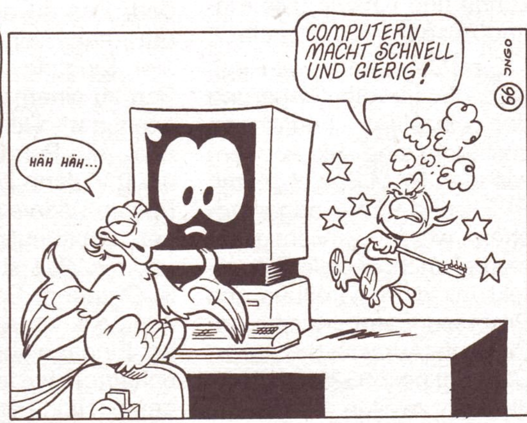
und starten Sie mit RUN.

200 – Zeilenende bei Eingabe
Zeiger auf das Eingabeende der logischen Zeile. Eine echte Zeile hat am Bildschirm 40 Zeichen. Um jedoch bei Programmen längere Eingaben zu ermöglichen, öffnet das Betriebssystem logische Zeilen mit 80 Zeilen Länge. Diese bestehen aus zwei untereinanderliegenden Zeilen am Bildschirm. In der Speicherstelle 200 steht die Position des letzten eingegebenen Zeichens.
201/202 – Zeiger auf Cursor-Zeile/-Spalte
wird bei GET und INPUT zum Auffinden der letzten Zeile und Spalte bei der Eingabe zu finden.
203 – gedrückte Taste
Tastencode der gerade gedrückten Taste. Laden Sie dazu
`LOAD"ADRESSE 203",8`
von der beiliegenden Diskette und starten Sie mit RUN.
204 – Flag für Cursor
Schalter für Cursor-Blinken. Hierzu befindet sich ebenfalls ein Demoprogramm auf Diskette. Sie laden es mit
`LOAD"ADRESSE 204/207",8`
und starten mit RUN. In diesem Programm wird mit GET eine INPUT-Routine simuliert.
205 – Zähler für Cursor
Blinkzähler für den Cursor. Wird durch den Interrupt des Betriebssystems 60mal in der Sekunde erhöht.
206 – Zeichen unter Cursor
Bildschirmcode des Zeichens unter dem Cursor.
207 – Blink-Flag für Cursor
gibt Aufschluß über den Anzeigemodus des Cursors:
0 = reverses, 1 = normales Zeichen.
208 – Flag Tastatur/Bildschirm (0)
Anhand dieser Flags unterscheidet das Betriebssystem, von welchem Eingabegerät es sich das Zeichen in den Arbeitsspeicher holt:
0 = Tastatur
209/210 – Pointer Start Bildschirmzeile
Zeiger auf die Adresse (Bildschirmspeicher), auf dem der Cursor gerade steht.
211 – Cursor-Spalte
enthält die Spaltennummer, an der sich der Cursor befindet. Laden Sie dazu mit
`LOAD"ADRESSEN 211/214",8`
von der beiliegenden Diskette und starten Sie mit RUN.
212 – Flag Hochkomma
Steht hier eine 0, befindet sich der C64 im »Gänsefüßchenmodus«, andere Zahlen bedeuten den Normalmodus.
213 – Länge Bildschirmzeile
Der Inhalt entscheidet, ob eine logische oder eine neue Zeile begonnen wird.
214 – Pos. Cursor-Zeile
enthält die Zeilennummer, an der sich der Cursor befindet (siehe auch 201).
215 – Speicher Tastencode in ASCII
Zwischenspeicher für den ASCII-Wert der zuletzt gedrückten Taste.
216 – Anzahl der Inserts
Pro Tastendruck auf <SHIFT INST/DEL> wird der Inhalt dieser Speicherstelle erhöht. Adresse 212 markiert den Hochkommamodus. Dann wird die Zeile ab dem Freiraum nach rechts verschoben. Bei jedem Tippen eines Zeichens wird der Inhalt wieder um 1 erniedrigt, bis mit 0 der Normalmodus wieder erreicht ist.
217 bis 242 – Tabellen der Bildschirmzeilen
Jede der 26 Speicherstellen enthält Angaben für die entsprechende Bildschirmzeile:
Bit 0 - Bit 3 = Speicherblock erstes Byte der Zeile
Bit 4 - Bit 7 = wenn »0«, dann zweiter Teil einer logischen Zeile.
243/244 – Pointer Farb-RAM
Zeiger auf Position des Cursors im Farbspeicher.
245/246 – Pointer ASCII (129/235 = 60289)
zeigt auf die Decodier-Tabelle für ASCII-Codewerte der Tasten.
247/248 – Pointer RS232 Eingabepuffer
zeigt auf den Beginn dieses Puffers. Immer wenn mit OPEN ein Kanal für die Gerätenummer 2 (RS232 am User-Port) geöffnet wird, werden 256 Byte am Ende des Basic-Speichers reserviert. Dieser Bereich überschreibt grundsätzlich schon gespeicherte Variablen. Daher muß bei Verwendung der RS232 in einem eigenen Programm als erstes (zum Beispiel mit OPEN2,2) der Kanal geöffnet werden.
249/250 – Pointer RS232 Ausgabepuffer
Zeiger auf den Anfang des RS232-Puffers im Gegensatz zu 247/248 für Ausgaben.
251 bis 254 – Frei
eignet sich für eigene Flags oder Register
255 – Zwischenspeicher
wird bei der Umwandlung von Fließkommazahlen in ASCII-Werte verwendet.

Mit der Kenntnis der Zero-Page ausgerüstet, lassen sich einige tolle Tricks ganz leicht programmieren. (gr)







Die Portbausteine des C64

Nicht nur ein Geheimdienst: CIA

Berühmt und berüchtigt. Häufig defekt und schwer zu beschaffen. Das sind die beiden 6526-Port-Bausteine des C 64. Aber wissen Sie auch, welche ungeahnten Möglichkeiten in diesen Bausteinen stecken?

von Bernhard Binz und Christian Mück

Im C 64 finden sich zwei ICs, über welche die CPU, die Hauptsteuereinheit des C 64, in Kontakt mit der Außenwelt tritt. Tastatur- und Joystick-Abfrage wären ohne diese Ein- und Ausgabe-Bausteine nicht möglich. Auch der serielle Bus, die Schnittstelle für Drucker und Diskettenlaufwerke, ist auf sie angewiesen.

Die im C 64 eingebauten Port-Bausteine stellen eine technische Weiterentwicklung der VIAs (Versatile Interface Adapter) 6522 dar, die im »kleinen Bruder« des C 64, dem VC 20, zur Verwendung kamen. Sie tragen den leicht zu Verwechslungen führenden Namen »CIA«, was aber hier für »Complex Interface Adapter« und nicht für »Central Intelligence Agency« steht. Die Typenbezeichnung dieser Bausteine ist 6526. Das weist darauf hin, daß diese Chips zum Anschluß an die Prozessoren der 65xx-Familie gedacht sind. Im C 64 ist der Prozessor ein 6510.

Öffnet man das Gehäuse des Computers, so sieht man in der linken oberen Ecke zwei 40polige ICs - die CIAs. An dieser Stelle eine Warnung an alle, die ihren Computer noch nicht länger als sechs Monate haben: Durch das Öffnen des Gehäuses kann jeder Garantieanspruch verlorengehen! Schauen wir uns zunächst die allgemeinen Eigenschaften der CIA6526 an. Sie besitzt zwei 8-Bit-Parallel-Ports mit den dazugehörigen Handshake-Leitungen, zwei programmierbare 16-Bit-Zähler (Timer), eine 24-Stunden-Echtzeituhr mit einer Auflösung von einer Zehntelsekunde und eine serielle Ein- und Ausgabe-Schnittstelle.

Eine CIA hat 16 8-Bit-Register, die für die Steuerung der einzelnen Funktionen zuständig sind. Sie können, wie der RAM-Speicher, mit PEEK und POKE angesprochen werden. Wichtig ist hierbei auch, daß die CIAs direkt mit dem Adreßbus des Prozessors verbunden sind. Die erste Adresse der ersten CIA liegt bei 56320 ($DC00), die der zweiten bei 56576 ($DD00). Will man eines der Register ansprechen, so addiert man zu der Basisadresse einfach die Nummer des Registers (0 bis 15) hinzu.

Die CIA 6526 besitzt zwei voneinander unabhängige 8-Bit-Ports (Schnittstellen). Für jeden dieser Ports existiert ein Register, welches die Zustände der einzelnen Port-Leitungen (Port A: Pin 2 bis 9, Port B: Pin 10 bis 17 in Abb. 1) Bit für Bit widerspiegelt. Liegt an einer dieser Leitungen Spannung an (High-Pegel), so ist das entsprechende Bit im Datenregister gesetzt. Liegt am Pin keine Spannung an (Low-Pegel), so ist das Bit in diesem Register gelöscht. Dieses Register heißt Datenregister. Um aber auch einen »bidirektionalen« Datenverkehr zu ermöglichen, gibt es zu jedem Port neben dem Datenregister noch ein Datenrichtungsregister (Port A: Register 2, Port B: Register 3).

Die CIA — ein Wunderwerk an Funktionen

Bidirektionaler Datenaustausch heißt, daß der Computer sowohl Daten über den CIA-Port, der am User-Port herausgeführt ist, ausgeben als auch empfangen kann.

Ist ein Bit im Datenrichtungsregister eines Ports gelöscht, so arbeitet das entsprechende Port-Bit als Eingang. Ein solcher Ausgang besitzt über einen hochohmigen Pull-up-Widerstand logischen High-Pegel. Um ein Port-Bit als Ausgang (+5 V) zu definieren, muß das entsprechende Datenrichtungsregister-Bit gesetzt werden. Das erklärt auch, warum man keinesfalls einen als Eingang geschalteten Port zu einem Ausgang machen darf. Denn angenommen, ein Port-Eingang läge auf 0 V, dann fließt über den Pull-up-Widerstand ein kleiner, unbedeutender Strom, und die Welt ist für den C 64 in Ordnung. Das kann sich aber schnell ändern, wenn der Eingang über das Datenrichtungsregister als Ausgang (High-Pegel) geschaltet wird. Denn dann ist ein Kurzschluß unvermeidlich, da die 5-V-Spannung jetzt direkt an Masse liegt und nicht, wie im Normalfall, über einen Widerstand.

Man kann also einen als Ausgang programmierten Port durch POKEn eines geeigneten Wertes in das Datenregister einen bestimmten elektrischen Zustand geben. Bei jedem Zugriff auf dieses Datenregister, egal ob PEEK oder POKE, erscheint am Pin PC (Pin 18 der CIA) ein kurzer Impuls, der dazu verwendet werden kann, dem Partner beim Datenaustausch, einem zweiten C 64 oder einem Drucker, mitzuteilen, daß Daten empfangen oder gesendet werden. Der Impuls dauert einen Systemtakt (etwa 1 μs). Der Pin PC von CIA 2 liegt am User-Port an Anschluß 8.

Die Ports der CIA werden zum Beispiel zur Datenübermittlung mit anderen Computern oder Peripheriegeräten verwendet. Möglich wäre auch eine Centronics-Schnittstelle, mit der man auch andere Drucker als Commodore-Drucker anschließen kann. Da der CIA-Port 8 Bit breit ist, lassen sich immer 8 Bit (1 Byte) gleichzeitig übertragen. Man spricht dann von einer 8-Bit-Parallelübertragung.

Wie bereits erwähnt, besitzt die CIA 6526 zwei programmierbare Timer. Das sind Zähler, die einen wählbaren Wert bis 0 dekrementieren (herunterzählen). Der Wert kann maximal 16 Bit groß sein, also 65535. Timer A belegt Register 4 (Low-Byte) und Register 5 (High-Byte), Timer B die Register 6 und 7. Auf einen Impuls hin, den der Fachmann »Trigger« nennt, erniedrigt der Timer den eingestellten Wert um 1. Der Trigger-Impuls kann für beide Timer von verschiedenen Quellen kommen. Es kann entweder der Systemtakt oder ein positives Signal am Pin CNT sein. »CNT« steht für »Count«. Dieser Pin ist für beide CIAs am User-Port herausgeführt (Pin 4 für CIA 1 und Pin 6 für CIA 2). Außerdem kann Timer B von Timer A getriggert werden, nämlich jedesmal, wenn dieser den Wert Null erreicht hat. Dadurch ist es möglich, aus den zwei Timern einen einzigen zu machen, der 32 Bit umfaßt, also von 0 bis 2^{32}-1 (= 4294967295) zählen kann.

Eingebaute Uhren: die Timer

Ein Erreichen negativer Werte nennt man Unterlauf. Bei jedem Unterlauf sieht der Timer in seinem Kontrollregister nach (Timer A: Register 14, Timer B: Register 15), ob das Bit 3 gesetzt ist. Wenn das der Fall ist, bleibt der Ti-

mer stehen. Diese Betriebsart nennt man »One Shot Mode«. Andernfalls beginnt der Timer erneut von dem Wert, der vorher eingegeben wurde, nach Null zu zählen (»Continuous Mode«). Starten und stoppen kann man den Timer jederzeit mit Bit 0 des jeweiligen Kontrollregisters. Setzt man das Bit, so startet der Timer, löscht man es, bleibt er stehen. Eine Übersicht über alle Betriebsarten gibt Tabelle 1. Die Timer können in konstanten Zeitabständen (bei jedem Unterlauf) einen Interrupt erzeugen, wenn ein bestimmtes Bit im Interrupt-Control-Register (Register 13, siehe auch Tabelle 1) gesetzt wird. Dieser Interrupt-Impuls, der von Timer A der CIA 1, im Normalfall jede 1/60-Sekunde ausgelöst wird, bewirkt einen Sprung ins Interrupt-Programm, mit dem unter anderem die Tastatur abgefragt und die Softwareuhr (TI$ und TI) gestellt wird.

Über die »Uhren« (TI,TI$) werden sich diejenigen, denen es auf hohe Genauigkeit ankommt, schon geärgert haben. Sie haben nämlich eine Ungenauigkeit von bis zu einer halben Stunde pro Tag. In der CIA ist jedoch eine Uhr mit hervorragender Genauigkeit und einer Auflösung von einer Zehntelsekunde.

Sehr ganggenaue Echtzeituhr

Diese Uhr wird durch die Netzfrequenz gesteuert (50 Hz), wodurch auch die hohe Präzision zu erklären ist. Man kann sogar eine Alarmzeit vorwählen, bei der die CIA einen Interrupt auslöst. Die Uhr belegt in der CIA die Register 8 bis 11 (Tabelle 1). Die Zeitwerte stehen in den Registern im BCD-Format. BCD ist die Abkürzung für »Binary Code Decimal«. Das heißt, daß jeweils 4 Bit (ein Nibble) einer Binärzahl zu einer Dezimalstelle zusammengefaßt werden. Dabei darf natürlich der Wert eines Nibbles nicht größer als 9 sein. Bei einer 8-Bit-BCD-Zahl ergibt sich also ein Höchstwert von dez. 99 (= bin 1001 1001) im Gegensatz zu dez. 255 bei einer reinen Binärzahl. Die Kombination 0001 0011 ergibt dezimal 13, denn 0001 ergibt 1 und 0011 ergibt 3. Bei Register 11 (Stunden) ist noch zu beachten, daß Bit 7, ähnlich wie bei einer Digitaluhr, das AM/PM-Flag (Vormittag/Nachmittag) darstellt. Es ist Nachmittag, wenn dieses Bit gesetzt ist. Sehen wir uns nun ein Beispiel für die Programmierung der Uhr an.

Nehmen wir an, die Uhrzeit soll auf 15:45:32,5 h gestellt werden. Der erste Schritt besteht darin, Bit 7 in Register 14 zu setzen (POKE 56589, PEEK(56589) OR 128). Hiermit wird die Uhr auf die Verarbeitung einer Netzfrequenz von 50 Hz eingestellt, anstelle der in USA üblichen 60 Hz. Bei einem Reset durchläuft der Computer eine Routine, die das Flag wieder auf 60 Hz umschaltet. Der zweite Schritt besteht darin, der CIA mitzuteilen, daß die Uhrzeit und nicht die Alarmzeit gesetzt werden soll. Dazu löscht man Bit 7 von Register 15 (POKE 56590, PEEK(56590) AND 127). Dieses Bit wird beim Einschalten und nach einem Reset auf »0« gesetzt (Uhrzeit setzen). Jetzt kann man beginnen, die Uhrzeit in die entsprechenden Register zu schreiben. Hierbei muß mit den Stunden begonnen werden, weil durch einen Schreibzugriff auf das Stundenregister die Uhr anhält. Das Register wird auf 15 Uhr gestellt, indem man die BCD-Zahl für 3 Uhr in das Register schreibt und zusätzlich Bit 7 auf Nachmittag setzt (PM). Es muß eingegeben werden: POKE 56587, 3:POKE 56587, PEEK(56587) OR 128 (oder POKE 56587), 131). Da die Uhr jetzt angehalten ist, können in aller Ruhe die Minuten, Sekunden und Zehntelsekunden gesetzt werden.

Zur Einstellung der Minuten ist anzugeben: POKE 56586, 39 (45 Minuten) für die Sekunden: POKE 56585, 50 (32 Sekunden). Für die Zehntelsekunden: POKE 56584, 5. Die Zehntelsekunden müssen als letzte angegeben werden, da durch einen Schreibzugriff auf dieses Register die Uhr erneut gestartet wird. Es ist also unbedingt nötig, nach einem Stellen der Uhr, auch wenn es einem nicht auf eine Zehntelsekunde ankommt, dieses Register zu beschreiben. Beim Setzen der Alarmzeit wird analog verfahren. Es muß jedoch Bit 7 von Register 15 gesetzt werden (POKE 56591, PEEK (56591) OR 128). Die Uhr wird natürlich beim Stellen der Alarmzeit nicht angehalten. Bei einer Übereinstimmung von Uhrzeit und Alarmzeit wird Bit 2 gesetzt und gegebenenfalls ein Interrupt ausgelöst. Ein Beispielprogramm dafür ist Echtzeituhr. Sie laden es von der beiliegenden Diskette mit
LOAD ''ECHTZEITUHR'', 8
und starten durch RUN.

Bild 1. Blockschaltbild der CIAs

Serielle Datenübertragung: Bit für Bit im Gänseschritt

Neben der parallelen Datenübertragung, die wir bereits im Abschnitt über den Daten-Port kennengelernt haben, gibt es noch eine andere Übertragungsart, bei der die einzelnen Bit eines Bytes nicht »nebeneinander«, sondern »hintereinander« übermittelt werden. Die Übertragungsart heißt seriell. Die Methode hat den Vorteil, daß man weniger Datenleitungen braucht, prinzipiell nämlich nur noch eine anstelle von in der Regel acht beim parallelen Datenverkehr. Der Nachteil ist aber die längere Zeit, die zur Übertragung der Daten nötig sind. Der Teil der CIA 6526, der für die serielle Datenübertragung zuständig ist, ist das serielle Datenregister (SDR, Register 12). Ob der Chip die Daten selbst senden oder empfangen soll, bestimmt Bit 6 des Kontrollregisters A (Register 14). Ist das Bit im Kontrollregister gesetzt, arbeitet das SDR als Ausgang, ansonsten als Eingang. Wenn das Register als Ausgang arbeitet, wird sofort, nachdem ein beliebiger Wert in Register 12 (serielles Datenregister) geschrieben wurde, damit begonnen, die 8 Bit dieses Werts nacheinander über den Pin 39 der CIA auszugeben. Dieser Pin ist für CIA 1 am User-Port über Anschluß 5 erreichbar, für CIA 2 an Anschluß 7. Der Timer A der CIA wird dazu benutzt, die Geschwindigkeit der Ausgabe festzulegen. Bei jedem zweiten Unterlauf des Timers (wenn er beim Herunterzählen eines Wertes in den negativen Zahlenbereich gelangt) wird ein Bit des Wertes, der im seriellen Datenregister steht, ausgegeben, wobei das höchstwertigste Bit (Bit 7) dieses Werts als erstes erscheint. Hierbei ist es natürlich notwendig, den Timer auf Continuous-Mode einzustellen, was durch Löschen des Bit 3

von Register 14 erreicht wird. Jedesmal, wenn ein Bit ausgegeben wurde, erscheint am Pin CNT ein kurzer Low-Impuls (also ein Spannungswechsel von +5 V auf 0 V und zurück), der einem möglichen Empfänger signalisiert, daß ein »neues« Bit zur Übergabe bereitsteht. Soll der Computer Daten empfangen, so muß Bit 6 vom Kontrollregister A (Register 14) gelöscht werden. Das SDR arbeitet dann als Eingang. Ein Bit wird immer dann in das Schieberegister übertragen, wenn am Pin CNT des empfangenden ICs ein Low-Impuls erscheint. Das höchstwertigste Bit wird hier, wie auch beim Senden, zuerst in das serielle Datenregister übernommen. Bit 3 des Interrupt-Control-Registers (Register 13) wird gesetzt, wenn das Schieberegister vollständig gefüllt oder geleert worden ist.

Mit den Schieberegistern, die im C 64 übrigens völlig unbenutzt sind, können wir zum Beispiel zwei Computer miteinander Daten austauschen lassen. Dazu sind die Basic-Programme »SENDEN« und «EMPFANGEN« mit auf Diskette. Die beiden C 64 müssen nur mit einem vierpoligen Kabel — wie in Abb. 2 gezeigt — verbunden werden. Der C 64, der Daten senden soll, muß das Sendeprogramm im Speicher haben, der Empfänger natürlich das Empfangsprogramm. Das Sendeprogramm arbeitet folgendermaßen: In Zeile 110 wird der Variablen CIA die Basisadresse der zweiten CIA zugeordnet. Dann wird die Timer-Rate festgelegt (Zeile 120 und 130), zuerst das Low- und dann das Highbyte. Hier wurde die schnellste Timer-Rate gewählt ($0002). In den folgenden drei Zeilen werden nacheinander von Register 14 (Kontrollregister A) Bit 3 gelöscht (Continuous-Mode), Bit 0 (Timer A startet) und Bit 6 gesetzt (Schieberegister arbeitet als Ausgang). In Zeile 170 wird der Daten-Port B als Eingang geschaltet, denn Bit 0 dieses Ports dient in unserem Programm als »Habe die Daten empfangen«-Leitung, über die der Empfänger dem Sender mitteilt, daß die Daten angekommen sind.

Datenübertragung über den User-Port

Danach werden in Zeile 180 und 190 die Startadresse, ab der die zu übertragenden Daten im Speicher stehen, und die Anzahl der Daten eingegeben. Jetzt kommt die Schleife, in der die Daten übertragen werden. In Zeile 210 wartet der Sender auf das Bereitschaftssignal des Empfängers. Dann wird der erste Wert aus dem Speicher geholt und in das Schieberegister gebracht. Um die eigentliche Übertragung kümmert sich dann die CIA. Nachdem der Empfänger wieder ein Freizeichen gegeben hat, wird der nächste Wert in das Schieberegister geschrieben und das Spiel beginnt von neuem.

Das Empfangsprogramm legt in Zeile 110 zunächst, wie auch das Sendeprogramm, die Basisadresse der zweiten CIA in der Memory Map des C 64 fest. Danach wird das Schieberegister auf Eingang geschaltet (Zeile 120) und Port B auf Ausgang (Zeile 130). Nun wird in Zeile 140 und 150 der Speicherbereich festgelegt, in dem die Daten gespeichert werden. Bei »Anzahl« muß natürlich die gleiche Zahl eingegeben werden wie beim Sender. In der folgenden Empfangsschleife wird als erstes ein Impuls auf der Bereitschaftsleitung erzeugt (Zeile 170-190). Die Schleife in Zeile 180 ist eine Verzögerungsschleife. Das Programm wartet in Zeile 200, bis alle 8 Bit eingetroffen sind, und speichert schließlich den empfangenen Wert ab (Zeile 210). Wenn beide Computer über den User-Port mit dem Kabel verbunden sind und sich die beiden Programme im Speicher befinden, kann es losgehen. Zum Test könnte man den Inhalt des Bildschirms übertragen, indem bei beiden Rechnern als Startadresse 1024 eingegeben wird und als Anzahl 1000. Es ist hierbei zu beachten, daß die Eingaben zuerst beim Senderprogramm abgeschlossen werden, denn sonst kann es passieren, daß der Sender das Bereitschaftssignal des Empfängers nicht mitbekommt und dann in der Schleife (Zeile 210) steckenbleibt. Die beiden Programme stellen jedoch nur eine Anregung zum Experimentieren mit den Schieberegistern dar. Es wäre zum Beispiel möglich, durch Verwendung von Maschinensprache die Übertragungsgeschwindigkeit zu erhöhen etc.

Statusanzeige der CIA

Am Inhalt des Interrupt-Control-Registers kann man bestimmte Ereignisse oder Zustände der CIA feststellen. Durch Schreiben in dieses Register kann man aber auch festlegen, welches Ereignis zusätzlich einen Interrupt auslösen soll. Jedem Bit dieses Register ist ein Ereignis zugeordnet.

Bit 0: Unterlauf von Timer A
Bit 1: Unterlauf von Timer B
Bit 2: Echtzeituhr, Alarmzeit erreicht
Bit 3: Schieberegister ist voll/leer, je nach Betriebsart
Bit 4: Low-Pegel am Pin FLAG aufgetreten.
Bit 5 und 6 sind unbenutzt.

Beim Interrupt-Control-Register (Register 13) muß man zwischen Schreib- und Lesezugriff unterscheiden. Schreibzugriff: Hierbei kann man die Interrupt-Quelle(n) bestimmen. Ist Bit 7 des Interrupt-Control-Registers gesetzt, lösen die Ereignisse, die über die Bits 0 bis 4 festgelegt wurden, einen Interrupt aus. Anders ist es, wenn Bit 7 des Interrupt-Control-Registers gesetzt ist, lösen die Ereignisse, die über die Bits 0 bis 4 festgelegt wurden, einen Interrupt aus. Anders ist es, wenn Bit 7 gelöscht ist. Dann werden nämlich alle Interrupts gesperrt, deren Bits gesetzt sind. Die Interrupts, deren Bits nicht gesetzt werden, bleiben in ihrem vorhergehenden Status. Soll zum Beispiel die Echtzeituhr beim Erreichen der Alarmzeit einen Interrupt auslösen, kann man folgende Befehle eingeben: POKE (Register 13), 31 (= bin. 00011111), wodurch sämtliche Interrupts zunächst gesperrt werden, weil es sein kann, daß vorher schon der eine oder andere Interrupt freigegeben wurde. Danach kann man den Alarmzeit-Interrupt durch POKE (Register 1), 132 (= bin. 10000100) freigeben. Lesezugriff: Beim

Reg.	Adresse	
0	56320 56576	($dc00) ($dd00)
1	56321 56577	($dc01) ($dd01)
2	56322 56578	($dc02) ($dd02)
3	56323 56579	($dc03) ($dd03)
4	56324 56580	($dc04) ($dd04)
5	56325 56581	($dc05) ($dd05)
6	56326 56582	($dc06) ($dd06)
7	56327 56583	($dc07) ($dd07)
8	56328 56584	($dc08) ($dd08)
9	56329 56585	($dc09) ($dd09)
10	56330 56586	($dc0a) ($dd0a)
11	56331 56587	($dc0b) ($dd0b)
12	56332 56588	($dc0c) ($dd0c)
13	56333 56589	($dc0d) ($dd0d)
14	56334 56590	($dc0e) ($dd0e)
15	56335 56591	($dc0f) ($dd07)

Lesen des Interrupt-Control-Registers erfährt man nun, welches Ereignis aufgetreten ist. Unabhängig davon, ob es als Interrupt-Anforderung zugelassen wurde. Ist ein Bit gesetzt, ist das zugehörige Ereignis aufgetreten. Zusätzlich ist Bit 7 gesetzt, wenn der Interrupt freigegeben war und ausgelöst wurde. Beim Umgang mit diesem Register ist jedoch Vorsicht geboten, da es beim Lesen gelöscht wird! Der Hauptunterschied der beiden CIAs im C 64 untereinander ist, daß CIA 1 (Basisadresse 56320) ausschließlich IRQs auslösen kann, CIA 2 (56576) nur NMIs, NMI und IRQ sind verschiedene Interrupt-Arten. Ein IRQ (Interrupt-Request) ist der Interrupt, der normalerweise alle 1/60-Sekunde vom Timer A der CIA 1 ausgelöst wird. Während des Interrupts wird beispielsweise die Tastatur abgefragt. Daher sollte man diesen Interrupt im Direktmodus nicht sperren. Der Computer würde das sofort mit einem Absturz quittieren. Ein NMI (Non-Maskable-Interrupt) wird beispielsweise auch durch die RESTORE-Taste ausgelöst, was aber nichts mit der CIA zu tun hat.

Die beiden Ports der CIA 1 (IRQ-CIA) werden zur Abfrage der Tastatur und von eventuell angeschlossenen Joysticks verwendet. Die Tastatur ist als eine 8 x 8-Matrix geschaltet (s. Zeropage S. 24). Ist die Tastatur abgeschaltet und statt dessen Joysticks in den Control-Ports eingesteckt, ist Port A für den Control-Port 2 zuständig, Port B zeigt den Zustand von Control-Port 1. Timer A gibt dem Prozessor den Abfrage-(Interrupt-)takt vor. Bei jedem Interrupt wird zum Beispiel die Tastatur nach der eben beschriebenen Methode abgefragt und die Softwareuhren TI und TI$ gestellt. Außerdem wird Timer A, zusammen mit Timer B, für Kassettenoperationen gebraucht. Timer B regelt, neben Kassettenoperationen, auch Zugriffe auf den seriellen Bus.

Die Echtzeituhr der CIA 1 wird vom Betriebssystem nur dazu benutzt, Zufallszahlen (RND(0)) zu erzeugen. Dies beeinflußt aber andere Programme nicht. Der serielle Port ist unbenutzt. Er steht dem Anwender am User-Port, Anschluß 5, zur Verfügung.

Die zweite CIA (NMI-CIA) hat weniger Aufgaben als die CIA 1. Bit 0 und 1 des Ports A dienen zur Auswahl der Videospeicherbank. Bit 2 hat nur im Zusammenhang mit einer RS232-Schnittstelle eine Aufgabe. Bit 3 bis 7 regeln den Datenverkehr auf dem seriellen Bus für Drucker und Diskettenlaufwerke. Die Bedeutung ist dabei folgende:
Bit 3: ATN out (Attention out)
Bit 4: CLK out (Clock out, Taktausgang)
Bit 5: SER out (Serial out, Datenausgang)
Bit 6: CLK in (Clock in Takteingang)
Bit 7: SER in (Serial in, Dateneingang)

Der Port B kann vollständig am User-Port benutzt werden. Er hat sonst keine Funktion. Die Anschlüsse sind C bis L. Die übrigen Teile der CIA 2 sind im C 64 unbenutzt und stehen dem Anwender ebenfalls zur freien Verfügung. Eine Ausnahme bilden hier die Timer A und B, die den Datenverkehr auf der RS232-Schnittstelle regeln. Probieren Sie einfach einmal aus, was Sie alles mit den CIAs anfangen können. Denn wie schon gesagt, es gibt einige Funktionen der CIAs, die der C 64 nicht nutzt. (gr)

Funktion	Verwendung
Datenregister Port A Ein gesetztes Bit signalisiert High an der entsprechenden Port-Leitung	Tastaturabfrage IEC-Bus + RS232
Datenregister Port B Wie Register 0, jedoch für Port B	Tastaturabfrage User-Port
Datenrichtungsregister Port A Ein gesetztes Bit programmiert die zugehörige Portleitung als Ausgang	zusammen mit Register 0 zusammen mit Register 0
Datenrichtungsregister Port B Wie Register, jedoch für Port B	zusammen mit Register 1 zusammen mit Register 1
Timer A, Low-Byte Beim Lesen wird der momentane Zählerstand erhalten, beim Schreiben der Zählerstand (Low-Byte) gesetzt, von dem der 16-Bit-Zähler nach Null zählt	IRQ (alle 1/60 s) RS232
Timer A, High Byte Wie Register 4, jedoch für High-Byte, Timer A Siehe auch Register 14 (Control-Register A)	zusammen mit Register 4 zusammen mit Register 4
Timer B, Low-Byte Wie Register 4, jedoch für Timer B Siehe auch Register 15 (Control-Register B)	für Kassetten Op. RS232
Timer B, High-Byte Wie Register 5, jedoch für Timer B Siehe auch Register 15 (Control-Register B)	zusammen mit Register 6 zusammen mit Register 6
Time of Day 1/10 Sekunden Bit 0-3 enthalten die 1/10 Sekunden im BCD-Format. Ist Bit 7 in Register 15 gesetzt, so wird beim Schreiben die Alarmzeit gesetzt, ansonsten die Uhrzeit. Bit 4-7 unbenutzt.	(für RND) unbenutzt
Time of Day Sekunden Dieses Register enthält die Sekunden im BCD-Format. Schreibzugriff siehe Register 8	(für RND) unbenutzt
Time of Day Minuten Dieses Register enthält die Minuten im BCD-Format Schreibzugriff siehe Register 8	(für RND) unbenutzt
Time of Day Stunden Bit 0-3 enthalten die Stunden im BCD-Format, Bit 4 die 10er Stunden. Bit 7 ist bei AM (vormittags) 0 und bei PM (nachmittags) 1. Bit 5+6 unbenutzt Schreibzugriff siehe Register 8	(für RND) unbenutzt
Serial Data Register (SDR) Schieberegister, über das Daten am Pin SP herausgeschoben und hereingeholt werden. Das höchstwertige Bit erscheint zuerst.	unbenutzt unbenutzt
Interrupt Control Register (ICR) Bit 0: Unterlauf Timer A Bit 1: Unterlauf Timer B Bit 2: Uhrzeit und Alarmzeit sind gleich Bit 3: Schieberegister voll oder leer (je nach Betriebsart) Bit 4: 1, wenn negative Spannungsflanke an FLAG aufgetreten ist Bit 5 und Bit 6 sind immer 0 Bild 7: Es stimmt mindestens ein gesetztes Bit im INT MASK und INT DATA-Register überein Achtung: Beim Lesen wird das ICR gelöscht!	
Control Register A (CRA) Bit 0: 1=Timer A starten 0=Timer A stoppen Bit 1: 1=Ein Umlauf von Timer A wird an PB 6 signalisiert, auch wenn dieses Port-Bit als Eingang programmiert ist. Bit 2: 1=Bei einem Unterlauf von Timer A wird PB 6 invertiert Bit 3: 0=Continous-Mode 1=One-Shot-Mode Bit 4: Wird eine 1 eingeschrieben, so wird Timer A sofort mit dem Wert geladen, der vorher in Register 4 + 5 stand, egal ob der Timer gerade läuft oder nicht. Bit 5: 1=Timer A zählt positive Flanken an CNT 0=Timer A zählt Systemtakte Bit 6: 0=Das Schieberegister ist Eingang 1=Das Schieberegister ist Ausgang Bit 7: 1=TOD verarbeitet 50 Hz Netzfrequenz 0=TOD verarbeitet 60 Hz Netzfrequenz	
Control Register B (CRB) Bit 0-4: entsprechen Bit 0-4 von CRA, jedoch für Timer B und PB 7 Bit 5+6 bestimmen paarweise die Triggerquelle 00=Timer B zählt Systemtakte 01=Timer B zählt positive Flanken an CNT 10=Timer B zählt Unterläufe von Timer A 11=Timer B zählt Unterläufe von Timer A nur, wenn CNT high ist Bit 7: 1=TOD Alarmzeit setzen 0=TOD Uhrzeit setzen	

Tabelle 1. Die Register der CIA

Bild 2. So können Sie zwei C 64 über die User-Ports koppeln. Verbinden Sie die angegebenen Anschlüsse.

Die Interrupts des Videocontrollers

Zeilenstop

Effekte, wie ein geteilter Bildschirm oder Grafik mit Zeichensatz gemischt, erstaunen immer wieder. Mit wenigen Grundlagen zur Interruptprogrammierung und den Beispiel-Tools erreichen auch Sie Ergebnisse, von denen Sie nie zu träumen gewagt hätten.

von Frank Riemenschneider

Von Dingen, deren Funktionsweise man nicht kennt, hat man schnell den Eindruck, daß sie zu kompliziert seien. Doch lassen Sie sich nicht zu sehr von kommerziellen Programmen verwirren. Wie immer wird auch hier nur mit Wasser gekocht. Mit ein bißchen Know-how über den C64-Videochip programmieren auch Sie in Zukunft diese Effekte. Der Videocontroller (VIC) des C64 ist für alles verantwortlich, was auf dem Bildschirm erscheint. Seine Hauptaufgabe ist es, den Bildschirm zu verwalten. Weiterhin ist er für den Aufbau sowie die Steuerung der Sprites, und eben alles, was mit Grafik zusammenhängt, verantwortlich. Um mit dem restlichen System kommunizieren zu können, enthält der VIC nicht weniger als 47 Register, deren genaue Belegung Sie bitte Tabelle 1 entnehmen. Dabei läßt sich jedes Register wie eine normale Speicherstelle ansprechen.

Die Basisadresse des Videocontrollers liegt bei 53248 ($D000). Analog zur CIA (Complex Interface Adapter oder Port-Baustein) besitzt auch der VIC mehrere Möglichkeiten, eine Systemunterbrechung oder eben einen »Interrupt« (Impuls am Pin IRQ) auszulösen. Dafür zuständig sind die Register 25 (IRR) und 26 (Interrupt-Mask-Register (IMR)). Beide Register haben dabei zur Vereinfachung die gleiche Bit-Belegung:

Bit 0=1: IRQ durch Rasterzeilen-Interrupt
Bit 1=1: IRQ durch Sprite-Hintergrund-Kollision
Bit 2=1: IRQ durch Sprite-Sprite-Kollision
Bit 3=1: IRQ durch Lightpen-Impuls
Bits 4-6: Unbenutzt
Bit 7=1: Mindestens eines der Bits 0 bis 3 ist gesetzt.

Soll nun eine bestimmte IRQ-Quelle festgelegt werden, ist das entsprechende Bit (0 bis 3) und das Bit 7 im IMR zu setzen. Im IRR vermerkt der VIC das Eintreten eines im IMR festgelegten Interrupts. Ist z. B. eine Sprite-Sprite-Kollision um IMR zugelassen, und tritt dieses Ereignis ein, setzt der VIC das Bit 2 im IRR. Ein IRQ wird also immer dann ausgelöst, wenn ein Bit sowohl im Register 25 (IRR) als auch im Register 26 (IMR) gesetzt ist. Mit anderen Worten immer dann, wenn das Ereignis im IMR als Interrupt-Quelle festgelegt wurde und es dann dieses Ereignis tatsächlich auch passierte.

Der grundsätzliche Umgang mit dem IRR und IMR soll anhand eines Beispiels verdeutlicht werden: Nehmen wir an, Sie wollen einen IRQ durch einen Rasterzeilen-Interrupt erlauben. Wie oben beschrieben, ist dazu das entsprechende Bit (hier Bit 0) und das Bit 7 zu setzen. Der Befehl lautet:

```
LDA #%10000001
STA IMR
```

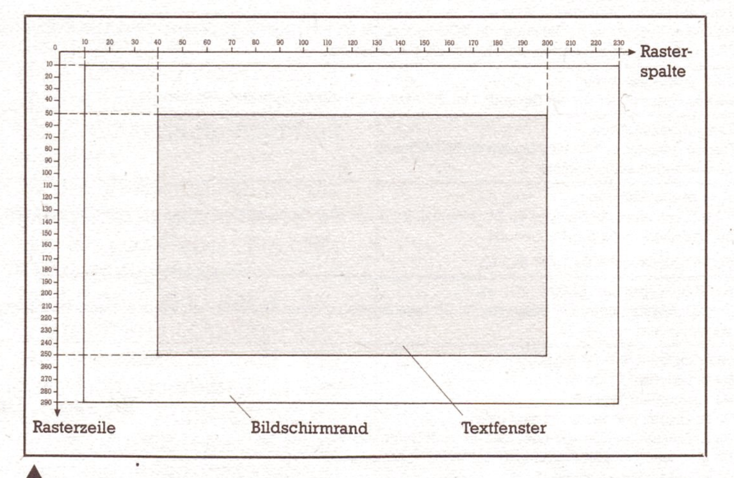
Interessant ist, daß die anderen Bits (1 bis 3) davon unbeeinflußt bleiben.

Nun wollen wir uns einmal das Gegenteil ansehen und sperren dazu den Rasterzeilen-Interrupt. Dies ist immer dann erforderlich, wenn vorher der IRQ durch Rasterzeilen erlaubt war. In diesem Fall muß ebenfalls das entsprechende »Quellen«-Bit gesetzt werden (hier wieder Bit 0), jedoch ist nun das Bit 7 zu löschen. Durch

```
LDA #%00000001
STA IMR
```

wird der IRQ durch Rasterzeilen-Interrupt verhindert.

Auch hier bleiben die nicht betroffenen Bits unangetastet (Bit=0).



[1] Durch Rasterzeilen und -spalten läßt sich die aktuelle Position des Elektronenstrahls bestimmen

Der wichtigste Hinweis jedoch betrifft das Register 25 (IRR); nach Auslösen eines IRQ wird es nicht gelöscht. Beispielsweise wird nach Verlassen einer durch eine Sprite-Kollision ausgelösten IRQ-Routine sofort wieder ein IRQ ausgelöst, da das Bit 2 nicht gelöscht ist. Dies muß daher unbedingt vom Programmierer in der IRQ-Routine vollzogen werden. Das Löschen geschieht durch einfaches Auslesen und Zurückschreiben des Registers:

```
LDA IRR
STA IRR
```

Mit entsprechenden Maschinenbefehlen wie AND, OR und CMP läßt sich feststellen, durch welches Ereignis der IRQ ausgelöst wurde. Diese Überprüfung muß stattfinden, da gleichzeitig mit den Interrupts des Videocontrollers auch noch der Systeminterrupt (Tastaturabfrage etc.) auftreten kann. Weil beide IRQs denselben Vektor benutzen, ist durch Testen der einzelnen Bits im IRR festzustellen, wodurch der IRQ ausgelöst wurde. Sonst kann es z.B. passieren, daß eine Explosionsroutine, die für eine Sprite-Kollision vorgesehen war, plötzlich einen System-Interrupt auslöst.

Wie oben schon erwähnt, ist der VIC in erster Linie für den Bildschirmaufbau verantwortlich. Um ein vernünftiges und vor allen Dingen flimmerfreies Bild auf dem Monitor zu erzeugen, sendet der Videocontroller pro Sekunde 25 Bilder. Jedes einzelne Bild setzt sich aus 625 Zeilen zusammen, die von einem Elektronenstrahl auf dem Bildschirm zum Leuchten angeregt werden. Dieser Elektronenstrahl wird vom Videocontroller in sog. Rasterzeilen und Rasterspalten eingeteilt, so daß jeder Bildpunkt praktisch eine Koordinate darstellt, durch die man die einzelnen Punkte genau festlegen kann. In Abb. 1 sehen Sie die genaue Einteilung des Bildschirms in Ra-

sterzeilen und Rasterspalten. Dabei entspricht die Auflösung des Textfensters der Punktauflösung einer Multicolor-Grafik (160 x 200 Punkte).

In diese Vorgänge kann man softwaremäßig eingreifen: Durch Auslesen des VIC-Registers 18 wird die gerade vom VIC aufgebaute Rasterzeile festgestellt. Da jedoch nur 8 Bit zur Verfügung stehen, also Werte bis max. 255, wird das fehlende neunte Bit durch Bit 7 des VIC-Register 17 ersetzt. Ein Problem stellt dabei die Geschwindigkeit des Rasterstrahls dar. Eine Rasterzeile baut sich in 178 Mikrosekunden auf - etwa 175 Taktzyklen. Die Ausführungszeit eines Assembler-Befehls liegt zwischen zwei und sieben Taktzyklen. Mit dieser Zyklenanzahl ist der C64 ausschließlich damit beschäftigt, das Register 18 abzufragen, um auf eine bestimmte Rasterzeile zu warten. Von Basic aus ist diese Abfrage natürlich überhaupt nicht möglich. Die Rasterspalte können wir aus Geschwindigkeitsgründen auch nicht auslesen, da jeder Punkt in weniger als einer Mikrosekunde aufgebaut wird und sich damit auch nicht durch die extrem schnelle Maschinensprache behandeln läßt. Aber auch hierfür bietet der VIC ein Rezept:

Beim Aufbau jeder Rasterzeile läßt sich ein IRQ auslösen. Dafür schreiben Sie zunächst die Rasterzeile, bei der ein IRQ ausgelöst werden soll, in das Register 18 (Low-Byte) und in das Bit 7 des Register 17 (High-Byte). Danach erlauben Sie den Rasterzeilen-IRQ durch Setzen von Bit 0 und 7 im IMR-Register. Ab jetzt wird jedesmal, wenn die von Ihnen bestimmte Rasterzeile aufgebaut wird, ein IRQ ausgelöst - die verblüffendsten Effekte lassen sich dadurch erzielen. Beispielsweise der »geteilte« Bildschirm:

Hier wird der Zeichensatz nach Erreichen einer bestimmten Rasterzeile wechselweise zwischen Groß- und Kleinschriftmodus umgeschaltet. Als Effekt läßt sich in der oberen Bildschirmzone ein anderer Zeichensatz darstellen als in der unteren. Denkbar ist auch der Wechsel auf einen hochauflösenden Grafikschirm. Oder es lassen sich durch Rasterzeilen-Interrupts mehr als acht Sprites gleichzeitig auf den Bildschirm zaubern.

Doch nun zu unserem kleinen Programm »Rasterzeilen-Interrupt«. Es liegt für Sie auf der beiliegenden Diskette bereit:

Als Quelltext für Assembler-Freaks, gespeichert unter dem Namen »RASTER-IRQ« (Abb. 2).

Das funktionsfähige File läßt sich nach der Anpassung des Quelltextes an Ihren Assembler erzeugen. Die Startadresse ist $C000.

In der Initialisierungsroutine (ab Zeile 26) wird der IRQ-Vektor verbogen und danach die Rasterzeile für den ersten IRQ festgelegt. Anschließend wird der IRQ durch Rasterzeilen zugelassen. Die neue IRQ-Routine (ab Zeile 44) löscht zuerst das IRR und prüft gleichzeitig, ob der IRQ wirklich durch den VIC ausgelöst wurde. Dies erkennen wir am Negativ-Flag, das immer dann gesetzt ist, wenn durch das Bit 7 im IRR ein Interrupt signalisiert wurde. Ist das Negativ-Flag nicht gesetzt, stammt der IRQ nicht vom VIC. Dann wird das IRQ-Register der CIA 1 gelöscht, der IRQ freigegeben und zur

VIC-Basisadresse: 53248/$D000	
Register 0	X-Koordinate Sprite 0 Die Bits 0 bis 7 repräsentieren die X-Koordinate von Sprite 0. Das oberste, neunte Bit wird dagegen im Register 16 gespeichert.
Register 1	Y-Koordinate Sprite 0 Die Bits 0 bis 7 repräsentieren die X-Koordinate von Sprite 0. Da diese Koordinate nicht größer als 255 werden kann, existiert kein neuntes Bit
Register 2	X-Koordinate Sprite 1 (Aufbau wie Register 0)
Register 3	Y-Koordinate Sprite 1 (Aufbau wie Register 1) und so weiter für acht Sprites
Register 16	High-Byte der X-Koordinate aller Sprites Bit 0: neuntes Bit von Sprite 0 Bit 1: neuntes Bit von Sprite 1 und so weiter Bit 7: neuntes Bit von Sprite 7
Register 17	VIC-Steuerregister 1 Bit 0 bis 2: Bildschirmverschiebung (Y) Bit 3=0: 24 Bildschirmzeilen =1: 25 Bildschirmzeilen Bit 4=0: Bildschirm aus =1: Bildschirm ein Bit 5=1: Standard-Bitmap-Mode Bit 6=1: Extended-Color-Mode Zugriff: READ Bit 7: neuntes Bit der aktuell aufgebauten Rasterzeile Zugriff: WRITE Bit 7: neuntes Bit der Rasterzeile, bei der ein IRQ ausgelöst werden soll.
Register 18	Rasterzeile (Low-Byte) Zugriff: READ Bit 0 bis 7 enthalten das Low-Byte der aktuell aufgebauten Rasterzeile Zugriff: WRITE Bit 0 bis 7 enthalten das Low-Byte der Rasterzeile, bei der ein IRQ ausgelöst werden soll.
Register 19	Lightpen X-Koordinate Bit 0 bis 7 enthalten die X-Koordinate des Bildschirmpunktes, der gerade aufgebaut wurde, als der Impuls vom Lightpen kam.
Register 20	Lightpen Y-Koordinate Funktion wie Register 19, jedoch für die Y-Koordinate
Register 21	Sprite ein/aus Bit 0=0: Sprite 0 aus =1: Sprite 0 ein und so weiter Bit 7=0: Sprite 7 aus =1: Sprite 7 ein
Register 22	VIC-Steuerregister 2 Bit 0 bis 2: Bildschirmverschiebung (X) Bit 3=0: 38 Zeichen pro Zeile =1: 40 Zeichen pro Zeile Bit 4=1: Multicolor-Modus Bit 5 bis 7: Ohne Bedeutung
Register 23	Spritevergrößerung X-Richtung Bit 0=0: Sprite 0 normal breit =1: Sprite 0 doppelt breit und so weiter Bit 7=0: Sprite 7 normal breit =1: Sprite 7 doppelt breit

VIC-Basisadresse: 53248/$D000	
Register 24	VIC-Basisadressen Bit 0: Ohne Bedeutung Bit 1 bis 3: Adreßbit 11 bis 13 des Zeichensatzes Bit 4 bis 7: Adreßbit 10 bis 13 des Video-RAMs Bit 14 und 15 für Zeichensatz und Video-RAM liegen invertiert in Bit 0 und 1 von Register 0/CIA2 (Adresse 56576/$DD00)
Register 25	Interrupt-Request-Register (IRR) Zugriff: READ Gibt Ursache für einen IRQ wieder: Bit 0=1: IRQ durch Rasterzeilendurchlauf Bit 1=1: IRQ durch Sprite-Hintergrund-kollision Bit 2=1: IRQ durch Sprite-Sprite-Kollision Bit 3=1: IRQ durch Lightpen-Impuls Bit 4 bis 6: Ohne Bedeutung Bit 7: Muß immer 1 sein, wenn mindestens eins der Bits 0 bis 3 gesetzt ist.
Register 26	Interrupt-Mask-Register (IMR) Zugriff: WRITE Gleiche Belegung wie Register 25. Hier kann der Programmierer wählen, wodurch ein IRQ ausgelöst werden soll.
Register 27	Sprite-Prioritäten Bit 0=0: Sprite 0 vor Hintergrund =1: Hintergrund vor Sprite 0 und so weiter Bit 7=0: Sprite 7 vor Hintergrund =1: Hintergrund vor Sprite 7
Register 28	Sprite-Multicolor Bit 0=0: Sprite 0 in Normalfarben =1: Sprite 0 in Multicolor und so weiter Bit 7=0: Sprite 7 in Normalfarben =1: Sprite 7 in Multicolor
Register 29	Spritevergrößerung Y-Richtung Bit 0=0: Sprite 0 normal hoch =1: Sprite 0 doppelt hoch und so weiter Bit 7=0: Sprite 7 normal hoch =1: Sprite 7 doppelt hoch
Register 30	Sprite-Sprite-Kollision Bei der Kollision zweier Sprites werden die entsprechenden Bits gesetzt, zum Beispiel Kollision Sprite 2 und 6: Die Bits 2 und 6 werden gesetzt. Zusätzlich wird das Bit 2 des Registers 25 gesetzt.
Register 31	Sprite-Hintergrund-Kollision Funktion wie Register 30, es wird jedoch Bit 1 im Register 25 gesetzt.
Register 32	Bildschirmrahmenfarbe
Register 33	Bildschirmhintergrundfarbe
Register 34 bis 36	Hintergrundfarben 1 bis 3
Register 37 bis 38	Sprite Multicolor-Farben
Register 39	Farbe Sprite 0
Register 40	Farbe Sprite 1 und so weiter
Register 45	Farbe Sprite 6
Register 46	Farbe Sprite 7

Tabelle 1. Alle Register ▶ des Videocontrollers

alten IRQ-Routine verzweigt. Die letzten Befehle werden Ihnen wahrscheinlich ein wenig merkwürdig vorkommen: Was hat die CIA 1 mit dem VIC zu tun, und warum wird der IRQ innerhalb eines Interrupts freigegeben? Wir werden diese Frage am Schluß ausführlich besprechen.

Die Rasterzeilen-Interrupt-Routine ab Zeile 58 wechselt den Zeichensatz und legt die Zeile, bei der ein IRQ ausgelöst werden soll, neu fest. Auf diese Weise erhalten wir einen dreigeteilten Bildschirm, wobei in dem ersten und letzten Drittel jeweils der Kleinschrift- und im mittleren Teil der Großschriftmodus aktiviert ist. Im Demoprogramm wird dieser Effekt durch den Programmteil »Raster-IRQ« verdeutlicht.

Der Rasterzeilen-IRQ ist eine der fantastischsten Möglichkeiten überhaupt.

Die nächste Art der IRQ-Auslösung ist die Kollision von Sprites untereinander oder mit einem Hintergrundzeichen.

Dies ist die Grundlage fast aller Spiele mit Sprites. So kann das Programm prompt auf Kollisionen, z. B. mit Farb- und Toneffekten, reagieren.

Wie oben schon erwähnt, müssen im IMR Bit1 und Bit 7 gesetzt sein, wenn der IRQ durch eine Sprite-Sprite-Kollision ausgelöst werden soll.

```
          .opt p4
 ;******** rasterzeilen-interrupt ********
          *=    $c000
rando     =     106
randu     =     194
irqalt    =     $ea31
raster    =     $d012
mask      =     $d01a
request   =     $d019
modus     =     $d018
klein     =     21
gross     =     23
 ;
 ;initialisierung
 ;===============
 ;
          sei             ;interrrupt verhindern
          lda  #<irqneu
          ldx  #>irqneu
          sta  $0314   ;irq-vektor auf neue
          stx  $0315   ;routine setzen
          lda  #rando
          sta  raster   ;1.zeile fuer irq
          lda  raster-1
          and  #%01111111   ;high-byte loeschen
          sta  raster-1
          lda  #%10000001   ;irq durch raster-
          sta  mask         ;zeilen festlegen
          cli             ;irq freigeben
          rts

 ;
 ;neue interruptroutine
 ;=====================
 ;
irqneu    lda  request    ;irq-register
          sta  request        ;loeschen
          bmi  rasterirq      ;zum raster - irq
 ;
 ;timer-interrupt
 ;===============
 ;
          lda  $dc0d          ;irq-reg. loeschen
          cli            ;irq zulassen
          jmp  irqalt         ;timer-irq-routine
 ;
 ;rasterzeilen-interrupt
 ;======================
 ;
rasterirqlda  raster   ;zeile holen
          cmp  #randu         ;unterer rand
          bcs  ok             ;ja, sprung
          lda  #klein         ;nein, auf kein-
          sta  modus          ;schrift schalten
          lda  #randu
          jmp  exit           ;zum schluss
ok        lda  #gross    ;grossschriftmodus
          sta  modus          ;einschalten
          lda  #rando
exit      sta  raster
          jmp  $ea7e          ;irq beenden
```

▲ **[2] Rasterzeilen teilen den Bildschirm in zwei unabhängige Bereiche**

Möchten Sie dagegen eine Sprite-Hintergrund-Kollision per Interrupt überprüfen, sind die Bits 2 und 7 zu setzen. Für den Fall, daß beide Kollisionsarten kombiniert werden, müssen Sie folglich Bit 1, 2 und 7 im IMR setzen. In unserem Demoprogramm »Sprite-IRQ« wird von dieser Möglichkeit Gebrauch gemacht. Auch hier befindet sich die assemblerfähige Version mit auf Diskette:

»SPRITE-IRQ« – der Quelltext zur eigenen Verwendung in Ihrem Assembler, allerdings ohne Spritedaten (Abb. 3).

Das fertige Maschinenfile, ergänzen Sie mit den Spritedaten. Nach der Anpassung an Ihren Assembler. Danach starten Sie es bei $C04D. Es handelt sich dabei um einen Ausschnitt aus einem Programm, bei dem zwei Ballonfahrer zusammenstoßen, wobei der eine Ballon explodiert (Farbeffekt) und abstürzt. Der heil gebliebene Ballon stößt jedoch auf dem Rückflug mit einem Haus zusammen und geht auch den Weg der Vorsehung. Ein, zugegeben, recht primitives Programm, was jedoch zeigt, worauf es ankommt.

```
          .opt oo,p4
 ;******** sprite-interrupt ********
          *=    $c04d
vic       =     $d000
irqalt    =     $ea31
mask      =     $d01a
request   =     $d019
 ;
 ;initialisierung
 ;===============
 ;
          sei             ;interrrupt verhindern
          lda  #<irqneu
          ldx  #>irqneu
          sta  $0314   ;irq-vektor auf neue
          stx  $0315   ;routine setzen
          lda  #%10000110   ;irq durch sprite-
          sta  mask         ;kollision festlegen
          cli             ;irq freigeben
          rts
 ;
 ;neue interruptroutine
 ;=====================
 ;
irqneu    lda  request    ;irq-register
          sta  request        ;loeschen
          bmi  raster         ;zum raster - irq
 ;
 ;timer-interrupt
 ;===============
 ;
          lda  $dc0d          ;irq-reg. loeschen
          cli             ;irq zulassen
          jmp  irqalt         ;timer-irq-routine
 ;
 ;rasterzeilen-interrupt
 ;======================
 ;
raster    lda  vic+31       ;spr-hintergr
          cmp  #00
          bne  back
 ;
 ;sprite-sprite kollision
 ;=======================
 ;
          ldx  #35
l1        ldy  #00
l2        tya
          sta  vic+39+1     ;sprite1 farbe
          eor  #15

          sta  vic+39+2     ;sprite2 farbe
          iny
          bne  l2
          dex
          bne  l1
l3        inc  vic+3   ;sprite1 absturz
          ldx  #13
lp1       ldy  #00
lp2       iny
          bne  lp2
          dex
          bne  lp1
          lda  vic+3
          cmp  #220
          bne  l3
          lda  vic+21
          and  #%11111101   ;sprite1 aus
          sta  vic+21
          lda  #00
          sta  vic+30       ;kollision loeschen
          jmp  $febc        ;irq beenden
 ;
 ;sprite-hintergrung kollision
 ;============================
 ;
back      ldx  #35
l4        ldy  #00
l5        tya
          sta  vic+39+2      ;sprite2 farbe
          iny
          bne  l5
          dex
          bne  l4
l6        inc  vic+5    ;sprite2 absturz
          ldx  #13
lp3       ldy  #00
lp4       iny
          bne  lp4
          dex
          bne  lp3
          lda  vic+5
          cmp  #220
          bne  l6
          lda  vic+21
          and  #%11111011    ;sprite2 aus
          sta  vic+21
          lda  #00
          sta  vic+31        ;kollision loeschen
          jmp  $febc         ;irq beenden
```

▲ **[3] Sprite-Sprite- oder Sprite-Hintergrund-Kollisionen wertet der Interrupt aus**

Die Initialisierungsroutine gleicht der des vorhergehenden Beispiels, wobei natürlich das IMR mit der Bitkombination für Sprite-Kollisionen geladen wird. Genau wie beim Rasterzeilen-IRQ prüft unsere neue IRQ-Routine zunächst, ob der IRQ vom Systeminterrupt oder VIC ausgelöst wurde und löscht das IRR. Wurde der IRQ durch den Systeminterrupt ausgelöst, wird wie beim Rasterzeilen-IRQ fortgefahren. Die Kollisionsroutine ab Zeile 66 sorgt durch Beschreibung der VIC-Register für die Abstürze der Ballone, wobei vorher ab Zeile 58 zwischen der Ursache Sprite-Sprite und Sprite-Hintergrund-Kollision unterschieden wurde.

Auch ein am Controlport 1 angeschlossener Lightpen löst durch einen Impuls einen Interrupt aus. Dabei wird jeder Punkt des Bildschirms mit Hilfe eines Elektronenstrahls zum Leuchten angeregt. Dieses Aufleuchten wird vom Lightpen registriert, der daraufhin einen Impuls an den Computer schickt. Natürlich ist dem VIC jederzeit bekannt, welchen Punkt

des Bildschirms er gerade aufbaut, so daß er sofort nach Erhalt des Impulses die aktuelle Rasterspalte und Rasterzeile speichern kann. Dies geschieht in den Registern 19 (Rasterspalte, x-Koordinate) und 20 (Rasterzeile, y-Koordinate). Der Programmierer kann nun durch Auslesen dieser Register die Koordinaten auswerten.

Genau wie bei den Sprite-Kollisionen tritt hierbei ein Problem auf:

Während des normalen Programmablaufs merken Sie nicht, ob der VIC einen Impuls vom Lightpen erhalten hat. Sie müßten praktisch ständig die Register 19 und 20 überprüfen, ob in diesen die Koordinaten eines neuen Punktes gespeichert sind, um entsprechend reagieren zu können. Zum Glück erlaubt aber auch hier der VIC einen IRQ. So können Sie mit der Auswertung des Lightpen-Impulses beginnen, sobald sich in den Registern 19 und 20 etwas geändert hat. Das Programm »LIGHTP.-IRQ« ist der Quelltext eines Programms, das auf einen solchen Impuls mit der wechselseitigen Umschaltung zwischen Grafik- und Textschirm reagiert (Abb. 4). Natürlich läßt sich das gebrauchsfertige Programm nach Anpassung aus dem Quelltext assemblieren. Danach liegt es im Speicher ab Position $C0DF.

Gestartet wird durch

```
SYS 49375
```

Schließen Sie bitte den Lightpen an. Haben Sie keinen, läßt er sich auch durch den Feuerknopf eines Joysticks ersetzen. Da der gleiche Anschlußpin beim Feuerknopf eines Joysticks verwendet wird, läßt sich dieser Impuls durch Knopfdruck simulieren. Die Initialisierungsroutine kennen wir schon von den vorhergegangenen Beispielen. Hier hat sie jedoch noch die Aufgabe, die hochauflösende Grafik einzuschalten. Wichtig ist nur die Zeile 47, in der im IMR das Bit 3 (für Lightpen-IRQ) und Bit 7 gesetzt wird. Die neue IRQ-Routine gleicht denen der letzten Beispiele. Ab Zeile 91 erkennen Sie aber eine Routine, die den Umschaltvorgang zwischen Grafik und Textschirm realisiert, indem die entsprechenden VIC-Register und das Register 0 der CIA 1 beeinflußt werden. Mit diesen Zeilen wird der Adreßraum für den VIC festlegt, der bekanntlich ja nur 16 KByte groß sein darf. Im Demoprogramm wird zunächst unsichtbar eine Sinuslinie auf den Grafikschirm gezeichnet. Durch Drücken des Feuerknopfs bzw. durch den Lightpen können Sie beliebig zwischen Text- und Grafikschirm hin- und herschalten.

```
         .opt p4
 ;******** lightpen/joystick-irq ********
         *=   $c0df
vic      =    $d000
flag     =    $9b
irqalt   =    $ea31
mask     =    $d01a
request  =    $d019
 ;
 ;initialisierung
 ;===============
 ;
         sei            ;interrrupt verhindern
         lda  #<irqneu
         ldx  #>irqneu
         sta  $0314     ;irq-vektor auf neue
         stx  $0315     ;routine setzen
         lda  #00            ;flag fuer text
         sta  flag           ;setzen
         lda  #<$6000
         sta  $71
         lda  #>$6000        ;grafikschirm
         sta  $72
         lda  #00            ;ab $6000
         ldx  #32
l1       tay            ;loeschen
l2       sta  ($71),y
         iny
         bne  l2
         inc  $72
         dex
         bne  l1
         lda  #<$4400        ;videoram ab
         sta  $71
         lda  #>$4400        ;$4400 mit farbe
         sta  $72
         lda  #110           ;fuellen- punkt-
         ldx  #04
l3       ldy  #00        ;fare hellblau,
l4       sta  ($71),y
         iny            ;hintergrund blau
         bne  l4
         inc  $72
         dex
         bne  l3
         lda  #%10001000  ;irq durch lightpen/
         sta  mask        ;joystick festlegen
         cli            ;irq freigeben
         rts
 ;

 ;neue interruptroutine
 ;=====================
 ;
irqneu   lda  request    ;irq-register
         sta  request       ;loeschen
         bmi  raster        ;zum raster - irq
 ;
 ;timer-interrupt
 ;===============
 ;
         lda  $dc0d         ;irq-reg. loeschen
         cli          ;irq zulassen
         jmp  irqalt        ;timer-irq-routine
 ;
 ;rasterzeilen-interrupt

 ;=====================
 ;
raster   lda  flag          ;hgr oder text
         cmp  #00
         beq  hgr           ;grafik einschalten
 ;
 ;auf textschirm schalten
 ;=======================
 ;
         lda  #%00011011
         sta  vic+17        ;grafik ausschalten
         lda  #%11001000    ;multicolor
         sta  vic+22        ;ausschalten
         lda  #%00010101    ;zeichensatz auf
         sta  vic+24        ;grosschrift
         lda  #%10010111    ;16 k-verschiebung
         sta  $dd00         ;des adressraumes
         lda  #00          ;flag auf hgr
         sta  flag         ;schalten
         jmp  $ea7e         ;irq beenden
 ;
 ;auf grafikschirm schalten
 ;=========================
 ;
hgr      lda  #%10111011
         sta  vic+17       ;grafik einschalten
         lda  #%11001000
         sta  vic+22       ;multicolor aus
         lda  #%00011101
         sta  vic+24       ;videoram nach $4400
         lda  #%10010110   ;16k-verschiebung
         sta  $dd00        ;des adressraumes
         lda  #01         ;flag auf text
         sta  flag        ;schalten
         jmp  $ea7e       ;irq beenden
```

▲ **[4] Das Lightpen-Steuerprogramm**

Wenn Sie die IRQ-Beispiele des VIC einmal vergleichen, sehen Sie, daß alle Routinen scheinbar eine merkwürdige Gemeinsamkeit aufweisen: Wenn sich herausstellt, daß der IRQ durch den Systeminterrupt ausgelöst wurde, wird das ICR der CIA 1 gelöscht und der IRQ mit dem CLI-Befehl freigegeben, obwohl wir uns in einer Interrupt-Routine befanden. Anhand des Rasterzeilen-Interrupts wollen wir diese Rätsel lösen.

Der Aufbau eines kompletten Bildschirms dauert $^1/_{25}$ s.

Da wir im Beispielprogramm während eines solchen Aufbaus zwei IRQs durch Rasterzeilen ausgelöst haben, bleibt zwischen je zwei IRQs eine Zeitspanne von etwa $^1/_{40}$ s. Gleichzeitig löst der C64 alle $^1/_{60}$ s einen Systeminterrupt aus. Dadurch lassen sich Überschneidungen der beiden IRQs auf Dauer nicht vermeiden.

Wir müssen auf jeden Fall einen IRQ auslösen, auch wenn gerade der System-Interrupt behandelt wird. Das bedeutet, daß dieser System-IRQ durch unseren VIC-IRQ unterbrochen werden muß, damit die einwandfreie Funktion der Rasterroutine gewährleistet ist. Dies ist durchaus zulässig:

Ist die normale IRQ-Routine unterbrochen, wird zunächst unsere VIC-Routine abgearbeitet und danach mit der Systeminterruptroutine fortgefahren. Wenn sie beendet ist, kann mit der Bearbeitung des ursprünglich unterbrochenen Programms weitergemacht werden. Dies klingt zwar kompliziert, ist aber eigentlich sehr einfach. Durch das Lesen des ICR der CIA 1 müssen Sie nur (wie bei allen Interrupts) die Ursache (Unterlauf Timer A) löschen. Dann wird mit dem CLI-Befehl der IRQ freigegeben.

Das heißt während ein Systeminterrupt abgearbeitet wird, tritt irgendwann ein IRQ durch Rasterzeilen auf. Um solche Überschneidungen zu verhindern, wird beim Auslösen eines IRQs automatisch das Interrupt-Flag gesetzt. Unser Rasterzeilen-Interrupt müßte also so lange warten, bis die Systeminterruptroutine beendet ist. Da in dieser Zeit der Bildschirmaufbau jedoch schon fortgeschritten ist, wäre die Folge eine unsaubere Trennung zwischen dem Klein- und Großschriftzeichensatz. Daher hat der IRQ vom VIC höchste Priorität:

Sie sehen, Interrupt-Programmierung ist nicht so kompliziert wie es uns die Freaks gerne erzählen. Es sind nicht einmal höhere Kenntnisse der Maschinensprache nötig. (gr)



Magazin im Überblick

Diese 64´er-Ausgaben bekommen Sie noch bei Markt & Technik für jeweils 6,50 DM, ab der Ausgabe 1/90 für 7,- DM, der Preis für Sonderhefte und Sammelboxem beträgt je 16,- DM. Tragen Sie Ihre Bestellung im Bestellcoupon ein und schicken Sie ihn am besten gleich los oder rufen Sie einfach an.

1/90: Gratis: BTX für alle! Mit Diskette im Heft / Joysticktest / Heimcomputer im DFU-Vergleich / Hurrican - die neue Spiele-Dimension

2/90: Systemvergleich: Die besten BTX-Dexoder / Funken mit dem C64 / Musik: "Power DIGI Editor" / 64er-Longplay "Oil Imperium"

3/90: Neue Speichertechniken / Grafikduell mit dem PC, Atari ST, Amiga und C64 / Neue Referenz: Brother M1826 !

4/90: Die Geos-Welt: das komplette Geos-System; Geos-Poster / Test Videofoy / Programm des Monats: Topprint

05/90: Listings des Monats: Sternwelt / Bauanleitung: Regelbares Dauerfeuer / Test Spielpack: Top oder Flop

6/90: Programmierung: endlich Basic 3.5 für C64 / Softwaretest: die besten Fußballprogramme / Videostudio, C64 in Börsenfieber

7/90: Extratouren: CD-Musicbox mit C64 und Bauanleitung Pulsmesser / Sammelposter C64 im Riesenforma

9/90: Großer C64-Reparaturkurs / Faszination: Amateurfunk / Neuigkeiten aus der Geos-Welt / Super-Spile zum Abtippen

10/90: Bauanleitungen: 5 Wochenend-Projekte / ECOM-das Super-Basic / Test: Die besten Drucker unter 1000 DM / C64-Reparaturkurs

11/90: Bausatztest: Der Taschengeldplotter / Vergleichstest: Drucker der Spitzenklasse / 5 Schnellbauschaltungen

12/90: Abenteuer BTX / Multitasking für C64 / Großer Spieleschwerpunkt / Programmierwettbewerb: 30 000 DM zu gewinnen

01/91: Die Besten Tips&Tricks / Neu: Reparaturecke / Floppy-Flop: Betriebssystem überlistet / Jahresinhaltsverzeichnis

02/91: Sensation: Festplatte für den C 64 / Drucken ohen Ärger / Listing des Monats: Actionspiel "Ignition" / Longplay: Dragon Wars

03/91: Bauanleitung: universelles Track-Display / Alles über Module für den C 64 / Festplatte HD 20 unter GEOS

Mit diesen Sammelboxen sind Ihre Ausgaben immer sortiert und griffbereit



Eine Sammelbox faßt einen vollständigen Jahrgang mit 12 Ausgaben und kostet 14,- DM. Bestellen Sie sie mit nebenstehendem Coupon.

Ab sofort können Sie auch telefonisch bestellen unter 089/20251527



Sonderhefte im Überblick

Die 64´er Sonderhefte bieten Ihnen umfassende Information in komprimierter Form zu speziellen Themen rund um die Commodore C 64 und C 128. Ausgaben, die eine Diskette enthalten, sind mit einem Diskettensymbol gekennzeichnet,

GRAFIK, ANWENDUNGEN, SOUND



SH 0020: Grafik
Grafik-Programmierung / Bewegungen



SH 0031: DFÜ, Musik, Messen-Steuern-Regeln
Alles über DFÜ / BTX von A-Z / Grundlagen / Bauanleitungen



SH 0045: Grafik
Listings mit Pfiff / Alles über Grafik-Programmierung / Erweiterungen für Amica-Paint



SH 0046: Anwendungen
Das erste Expertensystem für den C 64 / Bessere Noten in Chemie / Komfortable Dateiverwaltung



SH 0053: Das Beste aus 5 Jahren
10 Top-Programme aus allen Bereichen / PC-Simulationen aus dem C64



SH 0055: Grafik
Amica-Paint: Malen wie ein Profi / DTP-Seiten vom C64 / Tricks&Utilities zur Hires-Grafik

PROGRAMMIERSPRACHEN



SH 0056: Anwendungen
Gewinnauswertung beim Systemlotto / Energieverbrauch voll im Griff / Höhere Mathematik und C64



SH 0035: Assembler
Abgeschlossene Kurse für Anfänger und Fortgeschrittene



SH 0040: Basic
Basic Schritt für Schritt / Keine Chance für Fehler / Profi-Tools und viele Tips

FLOPPYLAUFWERKE, DATASETTE, DRUCKER



SH 0025: Floppylaufwerke
Wertvolle Tips und Informationen für Einsteiger und Fortgeschrittene



SH 0032: Floppylaufwerke und Drucker
Tips&Tools / RAM-Erweiterung des C64 / Druckerroutinen



SH 0047: Drucker, Tools
Hardcopies ohne Geheimnisse / Farbige Grafiken auf s/w-Druckern

C 64, C 128, EINSTEIGER



SH 0022: C 128 III
Farbiges Scrolling im 80-Zeichen Modus / 8-Sekunden-Kopierprogramm



SH 0026: Rund um den C64
Der C64 verständlich für Alle mit ausführlichen Kursen



SH 0029: C 128
Starke Software für C 128/ C 128D / Alles über den neuen C 128D im Blechgehäuse



SH 0036: C 128
Power 128: Directory komfortabel organisieren / Haushaltsbuch: Finanzen im Griff / 3D-Landschaften auf dem Computer



SH 0038: Einsteiger
Alles für den leichten Einstieg / Super Malprogramm / Tolles Spiel zum selbermachen / Mehr Spaß am Lernen



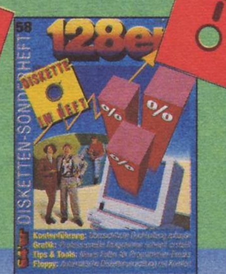
SH 0044: C 128
Grafikspeicher auf 64KB erweitern / Leistungstest GEOS 128 2.0 / Tips zum C 128



SH 0050: Starthilfe
Alles für den leichten Einstieg / Heiße Rythmen mit dem C 64 / Fantastisches Malprogramm



SH 0051: C 128
Volle Floppy-Power mit "Rubikon" / Aktienverwaltung mit "Börse 128"



SH 0058: 128er
Übersichtliche Buchhaltung zuhause / Professionelle Diagramme

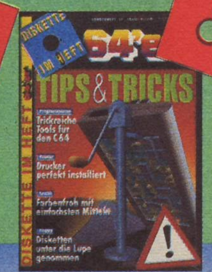
TIPS, TRICKS & TOOLS



SH 0024: Tips, Tricks& Tools
Die besten Peeks und Pokes sowie Utilities mit Pfiff



SH 0043: Tips, Tricks&Tools
Rasterinterrupts - nicht nur für Profis / Checksummer V3 und MSE / Programmierhilfen



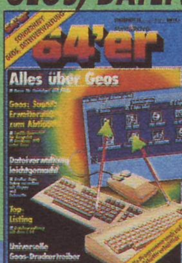
SH 0057: Tips & Tricks
Trickreiche Tools für den C64 / Drucker perfekt installiert

DTP

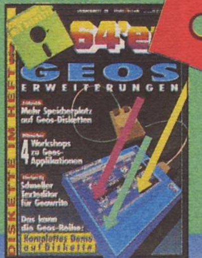


SH 0039: DTP, Textverarbeitung
Komplettes DTP-Paket zum Abtippen / Super Textsystem / Hochauflösendes Zeichenprogramm

GEOS, DATEIVERWALTUNG



SH 0028: Geos / Dateiverwaltung
Viele Kurse zu Geos / Tolle Geos-Programme zum Abtippen



SH 0048: GEOS
Mehr Speicherplatz auf Geos-Disketten / Schneller Texteditor für Geowrite / Komplettes Demo auf Diskette



SH 0059: GEOS
GeoBasic: Großer Programmierkurs mit vielen Tips & Tricks

SPIELE



SH 0030: Spiele für C 64 und C 128
Tolle Spiele zum Abtippen für C 64/C 128 / Spieleprogrammierung



SH 0037: Spiele
Adventure, Action, Geschicklichkeit / Profihilfen für Spiele / Überblick und Tips zum Spielekauf

SH 0042: Spiele
Profispiele selbst gemacht / Adventure, Action, Strategie



SH 0049: Spiele
Action, Adventure, Strategie / Sprites selbst erstellen / Virenkiller gegen verseuchte Disketten



SH 0052: Abenteuerspiele
Selbstprogrammieren: Von der Idee zum fertigen Spiel / So knacken Sie Adventures

SH 0054:
Adventures, Science Fiction, Horror / Viel Spaß mit "Rubberball"



SH 0060: Adventures
8 Reisen ins Land der Fantasie - so macht Spannung Spaß



SH 0061: Spiele
20 Heiße Super Games auf Diskette

BESTELLCOUPON

Ich bestelle die 64er Sonderhefte Nr. ______________

zum Preis von je
14,- DM (Heft ohne Diskette), 16,- DM (Heft mit Diskette)
24,- DM (nur für die Ausgabe SH 0051 + 0058)
Ich bestelle das 64er Magazin Nr. ______________

zum Preis von je
6,50 DM (bis Ausgabe 12/89), 7,- DM (ab Ausgabe 1/90)
.... Sammelbox (en) zum Preis von je 14,- DM
zzgl. Versandkosten
Ich bezahle den Betrag nach Erhalt der Rechnung

____
Name, Vorname

____
Straße, Hausnummer

____
PLZ, Wohnort

Schicken Sie bitte den ausgefüllten Bestellcoupon an:
Markt&Technik Leserservice, CSJ, Postfach 140 220, 8000 München 5, Tel.: 089/ 20 25 15 27

Programmieren ohne Fehler

Die Basic-Fah

Zwei Superprogramme prüfen Basic-Listings auf Eingabe- und Schreibfehler: »Syntax-Test« während der Programmierarbeit und »BKS« danach. Nervenaufreibende Fehlersuche – was war das?

von Nikolaus M. Heusler

Das »Basic-Kontrollsystem (BKS)« wurde ursprünglich entwickelt, um Leserlistings des 64er-Magazins auf Lauffähigkeit zu testen. Das Maschinenprogramm prüft ein im Speicher stehendes Basic-Programm Zeile für Zeile darauf, ob typische Fehler enthalten sind. Beispiele: eine Klammer zu viel oder zu wenig, Syntaxfehler, Formatfehler im Programmtext. Ganz besonders hat es »BKS« auf Sprungbefehle abgesehen. Hier werden alle Fehler gefunden, die möglich sind.

Sind Sie bereits fortgeschrittener Programmierprofi, kennen Sie das Problem: Sie haben ein komplexes Basic-Programm geschrieben, das von Assembler-Routinen unterstützt wird. Eifrig POKEt man im Speicher herum, wirft mit SYS-Befehlen um sich. Der Finger hat seinen Stammplatz am Resetschalter. Hoppla! Im Programmierstreß haben Sie einen falschen POKE-Befehl eingegeben, der das Basic-Programm angegriffen hat.

Prüfkriterien einstellen

Oder durch eine ungeschickte Manipulation wurde der Basic-Endezeiger $2D (45)/ $2E (46) so verändert, daß ein Start des Programms zum Totalabsturz führt.

Laden Sie die Check-Routine mit

```
LOAD"BKS 5.0(49152)",8,1
```

Geben Sie jetzt »NEW« ein, um alle Zeiger richtig zu stellen. Keine Angst, das Maschinenprogramm wird dabei nicht gelöscht. Jetzt muß man das Basic-Programm laden, das überprüft werden soll. Wenn Sie das vergessen haben, erscheint eine entsprechende Meldung. Der Inhalt des Basic-Anfangszeigers $2B (43)/ $2C (44) spielt keine Rolle.

Starten Sie nun das Kontrollsystem mit »SYS 49152«. Zunächst werden Sie gefragt, ob Leerzeichen bemängelt werden sollen. Manche Programmierer setzen zugunsten der Übersichtlichkeit in Basic-Programmen viele Leerzeichen (<SPACE>), andere verschmähen es. Haben Sie die Frage mit <J> beantwortet, testet »BKS« zusätzlich, ob außerhalb von Anführungszeichen und Data überflüssige Leerzeichen auftreten.

»BKS« kann auf Wunsch Sprünge (RUN, GOTO, GOSUB, THEN) finden, die auf eine REM-Zeile oder eine Trennzeile weisen (z.B.):

```
10 :
```

Problematisch kann es dann werden, wenn das Programm mit einem »unintelligenten« REM-Killer bearbeitet wird, der schonungslos alle REM-Zeilen »killt«. Diese Fehlermeldung kann man ebenfalls ausblenden.

Mit der folgenden Frage lassen sich zwei Fehler auf einmal übergehen: Nr. 7 taucht auf, wenn ein Sprungbefehl auf einen überflüssigen Strukturbefehl zeigt.

Beispiel:

```
10 A = 4 : GOTO 20
20 RETURN
```

Besser und einfacher ist diese Version:

```
10 A = 4 : RETURN
```

Der zweite vermeidbare Fehler ist Nr. 17. Er wird berücksichtigt, wenn der Programmierer direkt hinter das Befehlswort »THEN GOTO 12« schreibt. Kürzer ist: »THEN 12«. Da beides nur Schönheitsfehler sind, kann man ihre Ausgabe deaktivieren, wenn Sie die Frage mit <J> beantworten.

Die letzte der vier »Ausblendfragen« betrifft insgesamt drei Fehler. Wie Sie in unserer Tabelle 1 sehen, betreffen die Fehlernummern 12, 13 und 14 Befehlswörter, die ausgefuchste Programmierprofis nie verwenden: LET, NEW und STOP.

Anschließend werden Sie gefragt, ob die fehlerhaften Zeilen gelistet werden sollen bzw. ob die Ausgabe der Zeilennummern genügt. Die letzte Frage dient dazu, festzulegen, auf welchem Gerät die Fehlerliste ausgegeben werden soll. Drücken Sie eine der folgenden Tasten:

<S>: Die Fehlerliste erscheint auf dem Bildschirm. Dieser Modus ist vor allem bei kurzen Programmen angebracht oder als Test, ob die ausblendbaren Fehler zu oft vorkommen.

<D>: Die Liste wird auf dem Drucker ausgegeben. Die voreingestellte Geräteadresse ist 4, die Sekundäradresse 0. Die Routine wurde für Commodore-kompatible Drucker oder solche mit entsprechendem Interface geschrieben. Diesen Ausgabemodus sollten Sie wählen, wenn die Fehlerliste schriftlich vorliegen muß. Da dieselbe Routine wie die zur Bildschirmausgabe verwendet wird, umfaßt der Ausdruck nicht mehr als 40 Zeichen pro Zeile.

```
  220:  17[1] UEBERFLUESSIGER BEFEHL
  220 IFASC(Z$)=32THENGOTO270
  480:  17[1] UEBERFLUESSIGER BEFEHL
  480 V=I:FORX=1TOLEN(BE$):IFMID$(BE$,X,
1)=CHR$(160)ORMID$(BE$,X,1)=CHR$(32)THEN
GOTO490
  730:  20[1] UEBERFLUESSIGER BEFEHL
  730 IFZ$=""THEN711:IFZ$=""THEN710
11810:   2[1] FORMATFEHLER
11810 G$(11,15)= "
"
14390:  28[1] FORMATFEHLER
20046  GELOESCHT WERDEN, DANN ABS. LADEN
 UND MIT SYS50000 STARTEN

    ==== ENDE ====

    LEICHTE FEHLER: 5
    SCHWERE FEHLER: 0

READY.
```

[1] »BKS« war erfolgreich: Das DOC-File zeigt Programmierfehler en masse.

Fehlerdatei auf Diskette

<F>: Hier erfolgt die Ausgabe auf Diskette. Dabei wird ein Basic-Programm mit einem beliebigen Filenamen erzeugt, den Sie eingeben müssen. Normalerweise erhält er die Kennung »DOC« (für DOCument). Mit der Taste <CRSR links> läßt sich dieser Zusatz überschreiben (Vorsicht: Den Cursor nicht weiter als auf das »D« bewegen!). Diese Datei kann dann wie ein normales

nder

Basic-Programm geladen, gelistet oder mit RUN gestartet werden. Wie die Bildschirmausgabe dazu aussieht, zeigt Ihnen unsere Abbildung.

Nach Eingabe dieser Parameter geht »BKS« im Basic-Programm Zeile für Zeile auf Fehlersuche. Ist eine Zeile fertig, wechselt die Farbe des Bildschirmrahmens. Am Ende des Tests werden alle Dateien geschlossen und »BKS« beendet.

Wird »BKS« fündig, wenn also ein Fehler auftritt, gibt das Programm eine Zeile folgenden Formats auf dem Bildschirm, Drucker oder der Diskette aus:

```
AAA: BB[C] Hinweistext
```

»AAA« ist die Basic-Zeilennummer, die den Fehler enthält. Sie wird rechtsbündig ausgegeben. »BB« steht für die laufende Fehlernummer (s. Tabelle).

»C« zeigt den Fehlergrad:
C = 1: leichte Schönheitsfehler, die keine Fehlfunktionen des Programms auslösen. Beispiele sind überflüssige Leerzeichen, Listschutz, ein LET-Befehl usw.
C = 2: Hier verabschiedet sich das Programm mit einer Basic-Fehlermeldung. Beispiele: Sprungbefehle, die auf nicht vorhandene Zeilen zeigen, formal zerstörtes Programm (POKEs) oder Syntaxfehler.

Der hier mit »Hinweistext« bezeichnete Teil beschreibt die Art des Fehlers. Mögliche Texte sind: »FORMATFEHLER«, »SPRUNGFEHLER«, »ÜBERFLÜSSIGER BEFEHL« und »UNERLAUBTER BEFEHL«.

Da dieser Text nicht exakt die Fehlerart aufzeigt, ist der Parameter »BB« besonders wichtig: er kann 28 verschiedene Werte annehmen.

Das Kontrollsystem durchforstet das Basic-Programm Zeile für Zeile und Zeichen für Zeichen. Für die wichtigsten Befehle ist eine Routine vorhanden, die das Basic-Programm auf Fehler durchsucht. Tritt ein Fehler auf, springt das Check-Programm in eine Diagnoseroutine. Sie prüft zunächst, ob der Fehler ausgeblendet ist oder gezeigt werden soll.

Aufgrund dieser Besonderheit ist es nicht möglich, komplexere Fehler (»NEXT WITHOUT FOR« oder »RETURN WITHOUT GOSUB«) festzustellen.

Zusatzprogramm nennt Fehler beim Namen

»BKS« ist mit nützlichem Zubehör ausgestattet: dem Basic-Programm »BKS.WHAT«. Die Referenzliste, die »BKS« ausgibt, macht nicht viel Aufhebens in puncto Auskunft über die Bedeutung von Fehlern: Es erscheint lediglich Pauschaltext. »BKS.WHAT« kommentiert diese Liste und läßt sich einfach bedienen. Testen Sie das gewünschte Basic-Programm mit »BKS« und lassen Sie sich die Referenz auf Papier (!) ausgeben. Danach laden Sie, ohne vorher den Computer auszuschalten, das Zusatzprogramm mit

```
LOAD "BKS:WHAT 5.0",8
```

und starten es mit RUN.

Falls zuvor eine Auswertung mit »BKS« durchgeführt wurde, erscheint die Frage nach Datum und Namen des getesteten Basic-Programms. Alle anderen Angaben hat das Kontrollsystem bereits ans Kommentarprogramm übergeben. Nach kurzer Zeit gibt der Drucker einen ausführlichen Kommentar aus. Benutzt wird Sekundäradresse 7 (Groß-/Kleinschriftmodus). Das Druckprogramm wurde für MPS-Drucker geschrieben und enthält einige Steuercodes dieser Geräte. Mit entsprechendem Hardware-Interface ist es jedoch auch auf allen anderen herkömmlichen Druckern einzusetzen.

»Syntax-Test«, der zweite »Spürhund«, geht andere Wege: Er entdeckt Fehler bereits beim Eintippen eines Basic-Programms.

Im Gegensatz zu vielen anderen Computern überprüft der C64 bei der Eingabe einer Programmzeile

Fehler-Nr.	Bedeutung
1 (1)	Direkt nach der Zeilennummer folgt ein Nullbyte (dies wird zum Listschutz verwendet).
2 (1)	Im Programmtext kommt ein überflüssiges Leerzeichen vor. Nach DATA, in Anführungszeichen oder wenn diese Funktion abgeschaltet ist, werden Spaces nicht bemängelt.
3 (2)	Ein Befehl THEN, GOTO, LIST, RUN oder GOSUB zeigt auf eine nicht existierende Zeile (UNDEF'D STATEMENT).
4 (2)	Die hinter einem dieser Befehle stehende Zeilennummer ist größer als 63999.
5 (2)	Die hinter einem dieser Befehle stehende Zeilennummer enthält verbotene Zeichen (etwa »GOTO 4+6« oder »GOTO LABEL«). Gerade Pascal-gewohnte Programmierer werden diese Funktion des BKS zu schätzen wissen...
6 (2)	Eine Basic-Zeile ist länger als 255 Zeichen.
7 (1)	Ein GOTO- oder THEN-Befehl zeigt beispielsweise auf RETURN, GOTO oder RUN (Strukturierbefehl), den man einfach anstelle des Sprungbefehls hätte setzen können.
8 (1)	Ein Sprungbefehl zeigt auf eine REM- oder Trennzeile (das ist eine Zeile, die nur einen Doppelpunkt enthält). Das kann zu Problemen beim Abtippen führen, wenn die REM-Zeile weggelassen wird. Falls in einem Listing dieser Fehler zu oft vorkommt, kann auch die Ausgabe dieses Fehlers abgeschaltet werden.
9 (2)	Eine Basic-Zeile ist länger als 255 Zeichen (Fehler der Hauptroutine). Wenn dieser Fehler auftritt, dürfen eventuelle sonstige Fehler nicht mehr unbedingt ernstgenommen werden, da dann das System vollkommen durcheinandergerät.
10 (2)	Ein GOTO- oder RUN-Befehl zeigt auf sich selbst (nicht hinter THEN). Beispiel: `10 GOTO 10`
11 (2)	Der Befehl CONT darf nicht im Programmtext vorkommen.
12 (1)	Der Befehl STOP sollte nicht im Programmtext vorkommen.
13 (1)	Der Befehl NEW sollte nicht im Programmtext vorkommen.
14 (1)	Der Befehl LET sollte nicht im Programmtext vorkommen.
15 (1)	Hinter einem REM-Befehl steht ein geshiftetes »L« (Listschutz).
16 (2)	Ein illegales Token (Zeichen, Byte) kommt im Programmtext vor.
17 (1)	Der Befehl GOTO sollte nicht direkt hinter THEN stehen, einer von beiden genügt.
18 (2)	Hinter einem Befehl fehlt der Parameter (z.B. OPEN).
19 (2)	Hinter GO fehlt TO.
20 (1)	Hinter GOTO, RUN usw. folgen weitere Befehle, die niemals ausgeführt werden (z. B. »GOTO 20:PRINT "GEISTERBAHN"«)
21 (2)	Klammer(n) zu viel bzw. zu wenig.
22 (1)	Das Zeichen »↑« zur Potenzierung sollte vermieden werden (große Rechenungenauigkeit).
23 (2)	Der Befehl PRINT# wurde mit ?# abgekürzt (SYNTAX ERROR).
24 (2)	Falsche Reihenfolge der Basic-Zeilen. Das kann zu Problemen bei Sprungbefehlen führen.
25 (2)	Ein falscher Linkpointer kommt im Programmtext vor. (Vor jeder Basic-Zeile steht im Speicher ein Zeiger, der angibt, wo im Speicher die nächste Zeile beginnt. Anhand dieser Zeiger »hangeln« sich unter anderem die Sprungbefehle zur gesuchten Zeile.)
26 (2)	ON ohne legalen Sprungbefehl.
27 (2)	THEN ohne IF.
28 (1)	Der Pointer 45/46 zeigt nicht genau auf das Byte hinter dem Basic-Programm. Wahrscheinlich ist noch ein Maschinenprogramm angehängt, oder es wurde ein fehlerhafter RENEW-Befehl verwendet.

Die Fehlernummern des BKS und ihre Bedeutung (in Klammern der Härtecode, Parameter C) ▶

Fehlertest bei der Programmeingabe

nicht, ob diese syntaktisch richtig ist: Wurden alle Basic-Befehle exakt geschrieben? Hat man irgendwo eine Klammer vergessen? Steht nach einer INPUT-Anweisung ein Komma (statt des Semikolon)? Basic-Zeilen, in denen völliger Unsinn steht, lassen sich widerspruchslos speichern, RENUMBERn, listen und editieren. Erst nach dem Start mit RUN erscheint die unvermeidliche Fehlermeldung.

Die Erklärung ist einfach: Jedesmal, wenn eine Eingabe mit einer Ziffer beginnt, interpretiert der C64 dies als Zeilennummer und kümmert sich nicht mehr darum, ob der folgende Eingabetext richtig oder falsch geschrieben ist. Tippfehler werden bei der Eingabe nicht erkannt. Ein Beispiel:

```
10 PRENT "HALLO"
```

In dieser Basic-Zeile wurde die Anweisung »PRINT« falsch eingetippt. Nach dem Tastenbefehl <RETURN> übernimmt der Computer diese Zeile ohne zu murren.

»Syntax-Test«, die automatische Fehlerprüfung, befreit Sie von dieser Sorge. Nur wenn die eingegebene Zeichenfolge einwandfrei ist, wird sie wie jede normale Basic-Zeile behandelt und ins Programm übernommen. Andernfalls erhalten Sie den berühmten »Syntax Error« bereits jetzt, nicht erst nach dem Start mit RUN. Außerdem weigert sich der C64 beharrlich, die Programmzeile zu übernehmen. Der Fehler muß vorher ausgemerzt werden.

Das Utility wird geladen mit:

```
LOAD "SYNTAX-TEST",8
```

und mit RUN gestartet.

Nach dem Start verschiebt der C64 das Tool an den Speicherbereich ab $C000 (49152) und gibt die Einschaltmeldung aus. Gleichzeitig wird die NEW-Funktion ausgeführt, damit kein wertvoller Basic-Speicherplatz verlorengeht. Die Meldung »Ein« zeigt, daß die automatische Zeilenüberprüfung aktiviert ist. Befehle, im Direktmodus eingegeben, werden wie gewohnt ausgeführt. Gleichzeitig mit »BKS« ist »Syntax-Test« nicht lauffähig, da beide Programme denselben Speicherbereich des C64 benutzen.

»Syntax-Test« arbeitet mit allen Basic-Programmen einwandfrei zusammen, allerdings sollten diese nicht den Bereich von $C000 (49152) bis $C352 (50002) mit POKEs oder nachzuladenden Maschinenprogrammen belegen. Bei eingeschaltetem Checker kann es sonst zu Fehlfunktionen kommen. Hier sollte man ihn mit

```
SYS 49152
```

vorher deaktivieren. Die Meldung »SYNTAX-TEST AUS« erscheint: Der Basic-Interpreter des C64 arbeitet jetzt wieder in gewohnter Weise und nimmt auch fehlerhafte Zeilen an. Dieselbe SYS-Anweisung gilt auch zum Wiedereinschalten von »Syntax-Test«. Ein Basic-Programm, das sich gerade im Speicher befindet, wird davon nicht berührt.

Das Prüfprogramm leistet viel mehr, als allein die Schlüsselwörter zu »checken«: Es berichtet dem Computer über alles, das in der Zeile bzw. im Eingabepuffer ab $200 (512) steht. Der Checker führt quasi einen Probelauf der Basic-Zeile durch und erzeugt dabei die gleichen Fehlermeldungen wie der Interpreter des Basic 2.0 nach dem Programmstart mit RUN. Bei seiner Installation generiert »Syntax-Test« eine spezielle Version des Basic-Interpreters, die alle Befehle und Anweisungen nur »fast« ausführt, jedoch kurz vor der Aktion stoppt. Diese Methode bringt sehr viele Fehler ans Tageslicht: Falsche oder nicht vorhandene Parameter nach Anweisungen, fehlende Kommas und Klammern, Strings anstelle numerischer Ausdrücke (und umgekehrt) und die meisten »ILLEGAL QUANTITY ERRORS«.

»Run-Time-Errors«, die erst während des Programmablaufs erkannt werden, kann »Syntax-Test« jedoch nicht aufspüren, z.B. NEXT WITHOUT FOR, OUT OF DATA, DEVICE NOT PRESENT oder UNDEF'D STATEMENT. Der Checker untersucht lediglich, ob die Schreibweise (Syntax) der Basic-Befehle stimmt. Folgende seltsame Eingabe wird anstandslos verarbeitet:

```
LIST 20-40ABC
```

Der Computer listet die Zeilen 20 bis 40 und kümmert sich nicht um die Buchstaben, die noch in der Anweisung stehen. Oder:

```
POKE 2,200.23
```

ist völlig sinnlos, da der C64 nur ganzzahlige Werte in eine Speicherstelle schreiben kann. Trotzdem erscheint kein »ILLEGAL FRACTION ERROR«. »Syntax-Test« spürt nur solche Fehler auf, die auch Basic 2.0 auffallen würden.

So arbeitet der »Checker«

Obwohl größter Wert auf Kompatibilität gelegt wurde, kann es vorkommen, daß andere Utilities nicht mit »Syntax-Test« zusammenarbeiten. Die Zusammenarbeit mit anderen Hilfsprogrammen, die Zeropage-Vektoren ab $0300 (768) verbiegen, gelingt nur, wenn Sie diese Programme vorher laden und starten. Allerdings darf man nicht das RAM unter den ROMs benutzen. An diese Stelle kopiert »Syntax-Test« nach dem Programmstart nämlich das Basic- und Kernel-ROM und verändert es. Bei den meisten Anweisungen wird der Maschinenspracheteil zur Ausführung des Basic-Befehls nach der Parameterübergabe mit »RTS« abgezwackt. Andere Befehle, wie RESTORE oder END überprüft das Programm lediglich auf korrekte Schreibweise. Wenn Sie eine Basic-Zeile eingeben, springt das Programm über die geänderten Werte in den Vektoren $0302/$0303 (771/772) in eine neu programmierte Auswertungsroutine. Hier wird zwischen Basic-Anweisungen hinter Zeilennummern und dem Direktmodus unterschieden, bei dem die Kontrolle an die normale Basic-Interpreter-Schleife zurückgegeben wird. Dort wird die Anweisung wie gewohnt bearbeitet. Erkennt »Syntax-Test« eine Programmzeile, wird das modifizierte Basic aktiviert und die Zeile »gescannt«, und findet das Utility keinen Fehler, übernimmt sie der Computer.

Wenn Sie vorhaben, mit dem Basic des C64 eifrig zu programmieren, werden Sie auf diese beiden Utilities nicht mehr verzichten wollen. (bl)

Kurzinfo: Syntax-Test

Programmart: Utility zur Überprüfung von Basic-Eingaben
Laden: LOAD "SYNTAX-TEST",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: Fehler werden bereits beim Eintippen der Basic-Zeile erkannt
Benötigte Blocks: 4
Programmautor: Nikolaus M. Heusler

Kurzinfo: BKS (Basic-Kontroll-System)

Programmart: Kontrollprogramm für Basic-Listings
Laden: LOAD "BKS 5.0 (49152)",8,1
Starten: Nach dem Laden NEW eingeben. Anschließend Basic-Programm laden, das man überprüfen will. Aktivieren mit »SYS 49152«.
Besonderheiten: Fehlermeldungen werden wahlweise auf Bildschirm, Drucker oder Diskette ausgegeben.
Zusatzprogramm: »BKS.WHAT 5.0« gibt einen Kommentar zu Fehlermeldungen aus. Vor dem Aufruf sollte das Basic-Programm überprüft sein. Nach dem Test wird es mit
LOAD "BKS.WHAT 5.0",8
geladen und mit RUN gestartet. Das Programm ist an MPS-Drucker angepaßt.
Benötigte Blocks: 34
Programmautor: Nikolaus M. Heusler

Berührungspunkte

Cross-Referenz – Überblick für Programmierer

Basic-Programme zu dokumentieren, ist oft mühevoll, trotzdem aber sinnvoll. »Cross-Referenz« hilft, damit Sie nicht die Übersicht verlieren.

von S. Becker und J. Effenberg

Jeder Basic-Programmierer kennt das Problem: Langsam, aber sicher blickt man bei komplizierten und umfangreichen Listings nicht mehr durch. Oder versuchen Sie einmal, ein fremdes Programm auf Anhieb zu verstehen: Was bedeutet diese Variable, wo taucht sie sonst noch im Programm auf? Wie oft und an welcher Stelle wird diese Listingzeile angesprungen?

Bei gut strukturierten Programmen findet man sich schnell durch, doch die sind äußerst rar. Und selbst dann ist eine ausführliche Dokumentation äußerst hilfreich und erleichert den späteren Wiedereinstieg ins Programm.

Die Cross-Referenzliste unterstützt Sie beim Zusammenstellen einer Dokumentation. Gerade bei kommerziellen Softwareprojekten wird nicht darauf verzichtet. Sie enthält dabei nicht nur eine Aufzählung aller Sprünge (sofern bei höheren Programmiersprachen überhaupt vorhanden), sondern vor allem eine komplette Variablenliste. Exakt diese Funktionen besitzt unser Programm. Im einzelnen ist dies:

- eine Liste aller Zeilen, in denen Sprungbefehle enthalten sind. Angegeben werden Zeilennummer und aufgerufene Zeile,
- eine Liste aller Zeilen, die angesprungen werden. Angegeben sind Zeilennummer und alle Zeilen, die diese Zeile aufrufen,
- eine Liste aller im Programm vorkommenden Variablen (alphabetisch sortiert). Ausgegeben werden die Variablennamen und die Zeilennummern, in denen sie auftreten. Zusätzlich hat man die Möglichkeit, zu jeder Variablen einen kurzen Kommentar abzugeben,
- eine Liste aller Variablen mit dem Kommentar, aber ohne Hinweise auf die Zeilen, in denen sie vorkommen.

Mit diesen Informationen kann man viel anfangen. Wie sucht man nach Variablen? Wie nach Sprungbefehlen?

```
  PC   SR AC XR YR SP   NV-BDIZC
;C00B B0 C3 AF 00 F6   10110000
.M0800
:0800 00 0F 08 0A 00 99 20 22   ......."
:0808 50 52 4F 42 45 22 00 1C   PROBE"..
:0810 08 2C 01 BF 20 43 20 2D   .,.. C -
:0818 20 36 34 00 29 08 B0 04    64.)...
:0820 41 24 B2 22 41 3D 41 22   A$."A=A"
:0828 00 2F 08 50 C3 80 00 00   ./.P-...
.X
READY
LIST

10 PRINT "PROBE"
300 REM C - 64
1200 A$="A=A"
50000 END

READY.
```

[1] Unser Beispielprogramm, mit einem Maschinensprachemonitor als Hexdump betrachtet

Wenn Sie eine Basic-Zeile eintippen und anschließend <RETURN> drücken, legt sie der Interpreter im Speicher ab. Um nach bestimmten Befehlen und Anweisungen zu suchen, sollte man wissen, in welcher Form Basic-Zeilen abgelegt werden.

Basic-Programme im Speicher

Startadresse	Linkspointer	Zeilennummer	Basic-Text
$0801 2049	0F 08 15 8	0A 00 10 0	99 20 22 50 52 4F 42 45 22 00 153 32 34 80 82 79 66 69 34 0 Print » P R O B E «
$080F 2063	1C 08 28 8	2C 01 44 1	8F 20 43 20 2D 20 36 34 00 143 32 67 32 45 32 54 52 0 REM C - 6 4
$081C 2076	29 08 41 8	B0 04 176 4	41 24 B2 22 41 3D 41 22 00 65 36 178 34 65 61 65 34 0 A $ = » A = A «
$0829 2089	2F 08 47 8	50 C3 80 195	80 00 128 0 END
$082F 2095	00 00 0 0		

Tabelle 1. So legt der Basic-Interpreter Listingzeilen im Speicher ab

In den Adressen $2B/$2C (43/44) steht die Anfangsadresse des Basic-Programms in Form von Low- und High-Byte. Den exakten Wert erhält man durch folgende Formel:

```
PRINT PEEK(43)+256*PEEK
(44)
```

Diese Berechnung ergibt bei jedem Basic-Programm die Zahl »2049« ($0801 hexadezimal). Ab dieser Adresse beginnt der Interpreter, das Basic-Programm zu speichern. Untersuchen wir folgendes Listing:

```
10 PRINT "PROBE"
300 REM C - 64
1200 A$="A=A"
50000 END
```

Abb. 1 sowie Tabelle 1 zeigen, wie der Interpreter des Basic 2.0 diese vier Zeilen ablegt. Die Werte in den Klammern sind die Startadressen der einzelnen Basic-Zeilen.

Die beiden ersten Bytes (Low-, High-Byte) jeder Zeile nennt man Linkpointer (Verbindungszeiger) oder Koppeladressen. Der Wert dieser beiden Bytes entspricht immer der Startadresse der nächsten Basic-Zeile. In unserem Beispielprogramm ergeben die ersten beiden Bytes in der ersten Zeile die Zahl »2063« (15 + 8 x 256). Hier fängt die zweite Zeile an. Der Linkspointer der vierten Zeile (47 + 8 x 256) zeigt auf die Adresse »2095«. Die beiden Nullbytes in den Speicherstellen 2095 und 2096 signalisieren dem C64, daß das Programm an dieser Stelle zu Ende ist.

Byte 3 und 4 jeder Zeile enthalten (ebenfalls als Low- und High-Byte) die vom Programmierer vergebene Zeilennummer. In der ersten

Zeile ergibt dies die Zahl »10« (10 + 0 x 256).

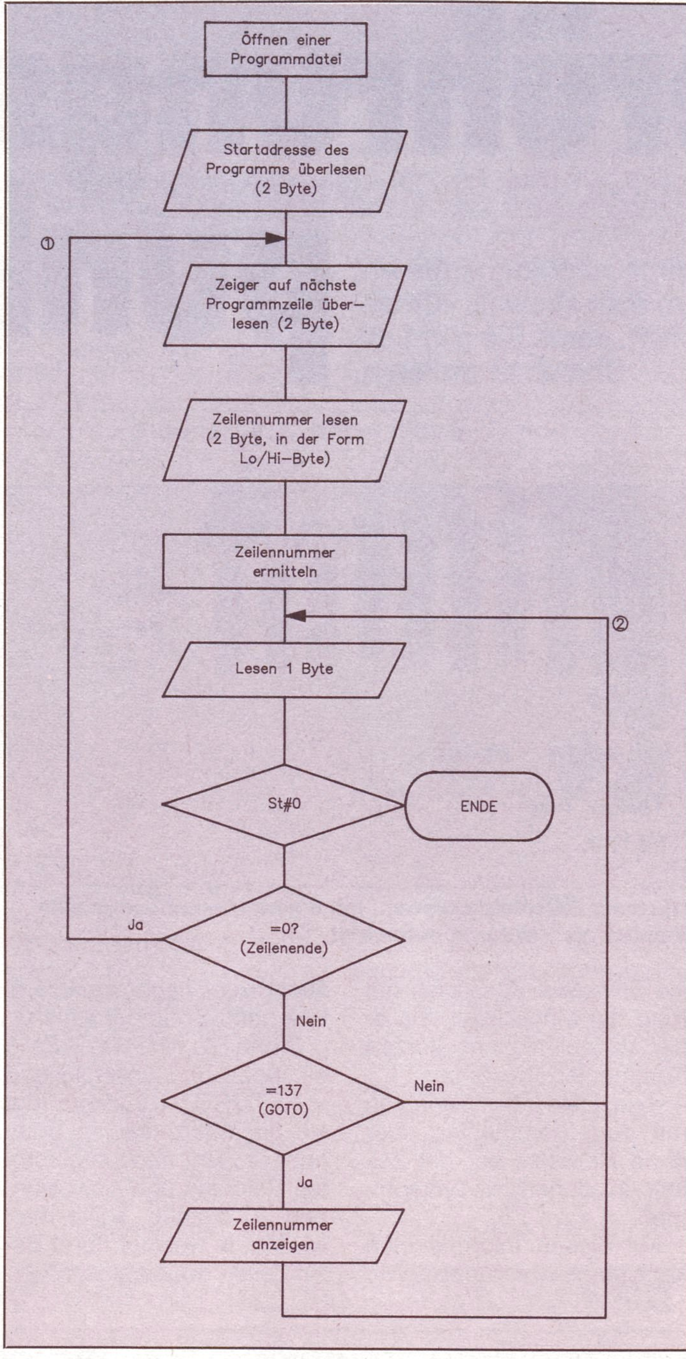
Nach der Linkadresse und der Zeilennummer folgt als 5. Byte das Token (Befehlskennbyte des Interpreters) für PRINT ($99). Um Speicherplatz zu sparen und einen schnellen Programmdurchlauf zu erzielen, werden sämtliche Basic-Befehle nicht als Klartext (PRINT, REM, END usw.), sondern als Kürzel in einem einzigen Byte (Token) abgelegt. Aus einer Tabelle im Interpreter-ROM holt sich der C64 die Information, welche Anweisung dem jeweiligen Token entspricht. Diese Tabelle liegt im Bereich von $A00C (40972) bis $A09C (41116). Tabelle 2 zeigt eine Liste aller Befehle des Basic 2.0 sowie deren Token.

Basic-Zeilen genau betrachtet

Ab dem sechsten Byte speichert der Computer ein Leerzeichen ($20), die Anführungsstriche ($22) und ASCII-Codes des Wortes »Probe«. Basic-Befehle oder -Anweisungen in Anführungszeichen speichert der Interpreter ebenfalls als ASCII-Code, nicht als Token. Deutlich wird dies im Speicherauszug der Zeile 300: Das erste Gleichheitszeichen (nach A$) stuft der Interpreter als Operator ein (Token $B2), das innerhalb der Anführungszeichen als normalen ASCII-Code ($3D). Eine Aufstellung aller ASCII-Werte finden Sie im Handbuch zum C64.

Das Ende jeder Programmzeile wird mit einer Null markiert ($00). Der Interpreter erkennt daran das Ende der Basic-Zeile und nimmt sich die nächsten beiden Bytes vor (Linkpointer zur nächsten Basic-Zeile). Enthalten diese ebenfalls Nullwerte, weiß der Interpreter, daß das Programm zu Ende ist. In unserem Beispiel sind es die Adressen $082F/$0830 (2095/2096).

Nahezu in der gleichen Form wird ein Programm auf Diskette gespeichert. Lediglich 2 Byte für die Ladeadresse des Programms kommen dazu. Beispielsweise beim absoluten Laden eines Programms (mit der Endung »,8,1«) holt sich der Basic-Interpreter diese 2 Byte, um festzustellen, an welcher Stelle im Speicher das Programm beginnt. Beim normalen LOAD-Befehl (Endung »,8«, also ohne die »1«), wird jedes Programm an den Basic-Anfang gesetzt ($0801, 2049).



[2] Nach diesem Prinzip durchsucht »Cross-Ref 64« Basic-Programme auf Diskette

An einem Beispiel wollen wir demonstrieren, wie man sich alle Zeilen anzeigen lassen kann, die den Befehl GOTO enthalten. Sehen Sie sich dazu das Flußdiagramm (Abb. 2) an.

Als erstes muß eine »Programmdatei« geöffnet werden:

```
OPEN2,8,2"NAME,P,R"
```

»P« steht hier für Programmdatei und »R« für READ (Lesen). Dann werden die Byte mit dem GET#-Befehl in den Computer geholt. Der INPUT#-Befehl ist zwar wesentlich schneller, »schafft« aber nur maximal 88 Zeichen, benötigt ein »Carriage Return« (CHR$(13) als Endekennzeichen (bei Programmen sind es Null-Byte) und verdaut keine Kommas, die speziell in Basic-Programmen oft vorhanden sind. Außerdem erlaubt uns der GET#-Befehl die Überprüfung einzelner Zeichen direkt nach dem Laden.

Lassen wir den C64 nun die ersten Byte eines Basic-Programms lesen.

In den ersten beiden Bytes steht die Startadresse. Bei Basic-Programmen lautet sie $0801 (2049). Darauf folgen 2 Byte für die Anfangsadresse der nächsten Basic-Zeilen (gemeint ist die Speicherplazierung der Adresse im Basic-RAM). Erst jetzt beginnt die »echte« Basic-Zeile: 2 Byte für die Zeilennummer (Low- und High-Byte), gefolgt vom Basic-Text. Das Ende einer Basic-Zeile erkennen Sie (und der Basic-Interpreter) an einem Null-Byte. Danach folgt wieder die Anfangsadresse der nächsten Basic-Zeile, der nächsten Zeilennummer usw. Das Ende eines Basic-Programms ist gekennzeichnet durch drei aufeinanderfolgende Null-Byte. Die Statusvariable ST erhält den Wert »64«.

Das war's schon. Damit sind Sie in der Lage, ein Basic-Programm komplett zu analysieren.

Referenz zum Basic-Programm

»Cross-Ref 64« arbeitet auf allen Commodore-Computern mit Floppy und Drucker. Laden Sie das Tool mit:

```
LOAD "CROSS-REF 64",8
```

Nach dem Start mit RUN kann die Ausgabe der fertigen Listen wahlweise auf einen Drucker oder den Bildschirm geleitet werden. Die Variable DV enthält die jeweils gültige Gerätenummer. Danach müssen Sie den Dateinamen des zu durchsuchenden Files eingeben. Verwenden Sie dazu als Beispiel unser Testprogramm »CROSS.DEMO« auf der Diskette zum Sonderheft.

Der C64 überprüft das Programm Byte für Byte, wie eine sequentielle Datei. Auf Diskette besitzt das zu testende Programm das gleiche Format wie im RAM. Le-

```
Programmname: test
sprungtabelle
------------------------------
zeile :     sprung auf zeile

  140:      1000   2000   3000

zeile :     wird angesprungen von

 1000:      140
 2000:      140
 3000:      140
```

[3] Die Sprungtabelle des Beispielprogramms »Cross.Demo«

```
Variablenliste ohne Zeilennummer
--------------------------------
a$(   :          eingebene Werte
aw    :    Parameter ON-Anweisung
aw$   :                 siehe aw
b(    :  dimensionierte Variable
c%(   :                     dto.
i     :            Laufvariable
j     :                     dto.
na$   :                Dateiname
x     :           Hilfsvariable
```

[4] Die Variablenliste von »Cross.Demo« ohne Kommentare

diglich die ersten beiden Bytes enthalten die Startadresse. Lautet diese anders als »2049« (z.B. bei Maschinenprogrammen), bricht das Programm die Bearbeitung ab.

Möchten Sie »Cross-Ref 64« mit einem anderen Commodore-Computer (z.B C 128) betreiben, laden und starten Sie das Programm »Startadresse« von der Diskette zum Sonderheft. Geben Sie nun den Namen des Programms ein, an deren Referenzlisten Sie interessiert sind. Die jetzt auftauchende Zahl übertragen Sie ins Listing von »Cross-Ref 64«, Zeile 290, und tauschen sie gegen den Wert »2049« aus. Beim C 128 z.B. heißt die Startadresse »7169«, beim C 16 »4097«.

Beim Durchsuchen »hangelt« sich der Computer durch die Zeilen. Trifft er auf das besagte Nullbyte ($00), beginnt für ihn eine neue Zeile. Beim ersten Durchlauf (Pass 1) überprüft »Cross-Ref 64« das Programm auf Token für die Sprungbefehle GOTO, RUN, GOSUB und THEN. Die Tokens besitzen die ASCII-Werte 137, 138, 141 und 167. Entdeckte Sprünge werden auf zwei Arten gespeichert: zunächst im Format

ZEILENNUMMER : SPRUNGZIEL,

darauf in der Darstellungsart

SPRUNGZIEL : ZEILENNUMMERN (Abb. 3)

»Sprungziel« ist immer die Zahl, die hinter GOTO oder THEN steht. »Zeilennummern« ist jeweils die Zeile, in der der Sprungbefehl auftaucht.

Die Zeilennummern werden in den Feldern PS$ und SP$ »konserviert«. Wird darin die maximale Länge von

```
liste der variablen       :
--------------------------

a$(  :    10    40     50   100
aw   :   130   140   1000  2000  3000
aw$  :   130
b(   :    10    20
c%(  :    10
i    :    20   100
j    :    30    40     50    60    90
na$  :    70    80
x    :    90  1000   2000  3000
```

[5] Eine Variablenliste ohne Zeilennummern läßt sich ebenfalls ausgeben

70 Zeichen überschritten, legt das Utility ein neues Feld an.

Hat das zu testende Programm den ersten Pass überstanden, wird die Liste auf das von Ihnen gewählte Gerät übertragen. Die aufsteigend sortierten Zeilennummern werden so ausgegeben, daß nie zwei gleiche untereinanderstehen. Statt dessen werden sechs Leerzeichen gedruckt.

Der zweite Durchlauf (Pass 2) verfolgt das Prinzip des ersten. Variablen mit zugehörigen Zeilennummern werden lediglich im Feld VA$ gespeichert und sortiert. Text hinter REM und DATA sowie Anführungszeichen überliest »Cross-Ref 64«.

In diesem Stadium kommt Arbeit auf Sie zu. Sie haben jetzt nämlich die Möglichkeit, die Variablen mit Text auszustatten. Dabei müssen Sie sich allerdings auf 25 Zeichen beschränken.

Anworten Sie auf die Frage »WOLLEN SIE ZU DEN VARIABLEN BEMERKUNGEN EINGEBEN« mit <J>, können Sie mit <CRSR aufwärts/abwärts> durch die Liste wandern. Die aktuelle Variable wird angezeigt. Haben Sie die richtige Stelle gefunden, drücken Sie <RETURN>. Nun läßt sich der gewünschte Text eingeben. Erneut RETURN-Taste tippen – das speichert den Text. Ein Tipp auf <N> läßt den Drucker losrattern. Diesen Vorgang kann man in anderen Variationen wiederholen (Abb. 4 und 5).

Basic-Befehl	Token	Basic-Befehl	Token
END	$80 (128)	SPC	$A6 (166)
FOR	$81 (129)	THEN	$A7 (167)
NEXT	$82 (130)	NOT	$A8 (168)
DATA	$83 (131)	STEP	$A9 (169)
INPUT#	$84 (132)	+	$AA (170)
INPUT	$85 (133)	-	$AB (171)
DIM	$86 (134)	*	$AC (172)
READ	$87 (135)	/	$AD (173)
LET	$88 (136)	)	$AE (174)
GOTO	$89 (137)	AND	$AF (175)
RUN	$8A (138)	OR	$B0 (176)
IF	$8B (139)	>	$B1 (177)
RESTORE	8C (140)	=	$B2 (178)
GOSUB	$8D (141)	<	$B3 (179)
RETURN	$8E (142)	SGN	$B4 (180)
REM	$8F (143)	INT	$B5 (181)
STOP	$90 (144)	ABS	$B6 (182)
ON	$91 (145)	USR	$B7 (183)
WAIT	$92 (146)	FRE	$B8 (184)
LOAD	$93 (147)	POS	$B9 (185)
SAVE	$94 (148)	SQR	$BA (186)
VERIFY	$95 (149)	RND	$BB (187)
DEF	$96 (150)	LOG	$BC (188)
POKE	$97 (151)	EXP	$BD (189)
PRINT#	$98 (152)	COS	$BE (190)
PRINT	$99 (153)	SIN	$BF (191)
CONT	$9A (154)	TAN	$C0 (192)
LIST	$9B (155)	ATN	$C1 (193)
CLR	$9C (156)	PEEK	$C2 (194)
CMD	$9D (157)	LEN	$C3 (195)
SYS	$9E (158)	STR$	$C4 (196)
OPEN	$9F (159)	VAL	$C5 (197)
CLOSE	$A0 (160)	ASC	$C6 (198)
GET	$A1 (161)	CHR$	$C7 (199)
NEW	$A2 (162)	LEFT$	$C8 (200)
TAB	$A3 (163)	RIGHT$	$C9 (201)
TO	$A4 (164)	MID$	$CA (202)
FN	$A5 (165)	GO	$CB (203)

Tabelle 2. Die Token der Basic-Befehlswörter

Programmhinweise

Die Druckerausgabe ist leicht zu modifizieren. Die Sekundäradresse 7 der OPEN-Befehle in den Zeilen 580 und 1180 arbeitet mit nahezu allen handelsüblichen Druckern zusammen, die seriell (z.B. über ein Interface) mit dem Computer verbunden sind. Eingestellt ist dabei die Klein- bzw. Großschrift.

Da Basic-Programmierer oft dazu neigen, bei wachsendem Umfang des Listings auf den berüchtigten »Spaghetti-Code« zurückzugreifen, wird man schnell die Vorteile dieses Programms zu schätzen wissen. (bl)

20-Zeiler-Parade der Superlative

Kleine Tools, große Wirkung

Wer sagt, daß gute Programme lang sein müssen? Unsere »20-Zeiler« bieten z.B. eine Erweiterung auf fast 50 KByte Basic-Speicher und eine Minitextverarbeitung!

Der kürzeste Zeichensatzeditor

Jeder bekommt mal Lust, den Original-Commodore-Zeichensatz zu ändern. Falls kein komfortabler Charakter-Editor zur Hand ist, laden Sie das Utility mit:

```
LOAD "ZEICHENSATZ-EDI",8
```

und starten Sie es mit RUN. Der Bildschirm zeigt das Editierfeld und die originalen Zeichenmuster (Abb. 1). Das Programm benutzt den Klein-Großschrift-Modus. Sie werden aufgefordert, den Bildschirmcode (!) des gewünschten Zeichens einzugeben. Das entsprechende Muster erscheint im Editierfeld. Folgende Tastenfunktionen stehen zur Verfügung:

<SPACE>: Bitpunkt setzen, löschen
<CLR>: Editierfeld löschen
<N>: neues Zeichen aufrufen
<L>: Zeichensatz laden
<S>: Zeichensatz speichern.

Neue Zeichenmuster (z. B. »Zeichensatzdemo« auf der beiliegenden Diskette) muß man ohne das Tool absolut laden (Endung »,8,1«) und mit **POKE 53272,27** einschalten. Vorsicht: Der Zeichensatz liegt ab Adresse $2800 (10240) im Speicher. Längere Basic-Programme könnten diesen Bereich überschreiben.

(Stephan Koch/bl)

Der bestversteckte Speicherbereich

Jeder weiß, daß unter dem ROM des I/O-Bereichs und des Betriebssystems immenser RAM-Speicherplatz brachliegt: 12287 Byte. Wie man sie nutzt, ist kaum bekannt. Normalerweise verwenden Programmierer diesen Bereich, um dort Text- oder Grafikdaten abzulegen. Diese Bytes zu sichern, stellt manchen Programmierer vor ein schier unlösbares Problem: Die SAVE-Routine $FFD8 (auch in Maschinensprache aktiviert) speichert immer nur die ROM-Daten, nicht das darunterliegende RAM. »Memsave« korrigiert diese Schwachstelle.

Laden Sie unseren 20-Zeiler mit:

```
LOAD "MEMSAVE.INSTALL",8
```

Nach dem Start mit RUN schreibt der Computer das Maschinensprache-File »MEMSAVE 828« auf eine Diskette im Laufwerk, auf der noch mindestens ein Block frei sein muß. Künftig muß das Programm absolut geladen werden:

```
LOAD "MEMSAVE 828",8,1
```

Möchten Sie im genannten Bereich ab $D000 (53248) RAM unterm ROM speichern, müssen Sie folgenden Befehl benutzen:

```
SYS 828, "Filename
", Startadres-
se, Endadresse
```

Wer z.B. das Simons-Basic-Modul besitzt, wird dieses Utility zu schätzen wissen: Diese bekannte Basic-Erweiterung legt Hires-Grafiken grundsätzlich ins RAM unters ROM ab $E000 (57344) - leider fehlt die entsprechende Speicherroutine, um die Grafik auf Diskette zu retten. Da »Memsave« im Kassettenpuffer liegt, treten mit »Simons Basic« keine Probleme auf.

(S. Kuhr/J. Aßmann/bl)

Der komfortabelste Notizblock

Wie geht's am einfachsten, Briefe, Mitteilungen und Notizen per Computer einzutippen? Von wegen umständlich ein Textverarbeitungsprogramm laden: Man schreibt einfach per Tastatur auf den Bildschirm! Aber: spätestens beim Sichern des Textes treten die ersten Schwierigkeiten auf. »MicroWriter+« kennt diese Probleme nicht: Es macht aus den Wörtern und Zeichen eine Diskettendatei, die man jederzeit wie ein normales Basic-Programm laden und starten kann.

Laden Sie das Utility mit:

```
LOAD "MICRO-WRITER+",8
```

Wenn Sie es mit RUN starten, müssen Sie den Filenamen eingeben, den der Text erhalten soll. Die Floppy bereitet die Datei auf Diskette vor. Nach wenigen Sekunden wird der Bildschirm gelöscht: Er steht bereit für Ihre Eingaben. Da das Programm nur im Großschriftmodus arbeitet, ist es unerheblich, ob Sie die SHIFT-Taste verwenden. Ausnahmen:

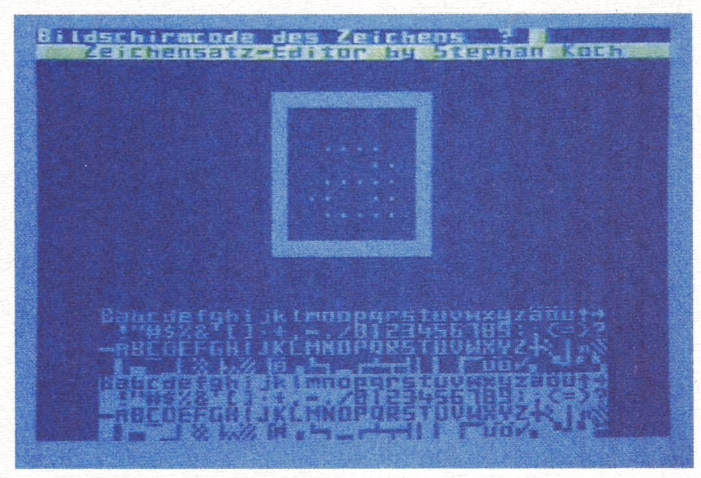
<SHIFT +>: Ä
<SHIFT ->: Ü
<CBM ->: Ö

Außerdem kann man mit <SHIFT A> bis <SHIFT F> grafische Effekte in den Bildschirmbrief einbauen. Mit <F8> speichern Sie den Bildschirmbrief. Sämtliche Verbesserungen und Cursor-Bewegungen werden mitgespeichert.

Wenn Sie die Briefdatei wie ein normales Basic-Programm laden und mit RUN starten, werden Sie einen geänderten Zeichensatz (kursiv) und ein akustisches Signal als Untermalung beim Textablauf bemerken. Das Beispielprogramm »MW.Demo« zeigt die veränderte Tastaturbelegung. Durch folgende Tasten läßt sich der Ablauf der Bildschirmnotiz steuern:

<F1>: langsam,
<F3>: Normalgeschwindigkeit (voreingestellt)
<F5>: schneller Ablauf.

(Ludger Bäumer/bl)



▲ **[1] Schnell und unkompliziert neue Zeichen entwerfen: Zeichen-Edi**

Der schnellste Basic-Schnüffler

Dieses Tool durchsucht Basic-Programme nach einer gewünschten Zeichenkette (String). Voraussetzung ist, daß der gesuchte Ausdruck in Anführungszeichen in dem Programm steht, das durchforstet werden soll. Außerdem kann das Utility gefundene Zeichenketten durch andere ersetzen. Der neue String muß dieselbe Länge wie der alte besitzen, sonst erscheint die Fehlermeldung »String too long error«.

Laden Sie den 20-Zeiler mit:

```
LOAD "FIND/REPL.BAS",8
```

und starten Sie das Programm mit RUN. Achten Sie darauf, daß eine Diskette mit mindestens zwei freien Blöcken im Laufwerk liegt. Nach dem Einlesen der Daten werden Sie gefragt, ob Sie die Maschinenspracheдatei »Find/Replace« auf Diskette speichern möchten. Dieses File finden Sie bereits auf der Diskette zu diesem Sonderheft. Künftig laden Sie das generierte Maschinenprogramm mit:

```
LOAD "FIND/REPLACE",8,1
```

Nach der Eingabe von »NEW« ist es unnötig, das Tool mit einem SYS-Befehl zu initialisieren. Holen Sie jetzt das Basic-Programm in den Speicher, das durchsucht werden soll. Der »Forschungsauftrag« lautet:

```
SYS 50000,"Suchstring",
"Ersatzstring", Anfangs-
zeile, Endzeile
```

Die Angaben von Ersatzstring sowie Anfangs- und Endzeile können entfallen. Als Joker im Suchstring dürfen Sie Fragezeichen <?> an beliebiger Stelle einsetzen.

Beim Start kopiert das Utility das Basic-ROM ins RAM und manipuliert diverse Interpreter-Routinen, u. a. die des LIST-Befehls (kein Ausstieg mehr bei SHIFT-L!). (Roland Ulber/bl)

Die kürzeste Textverarbeitung

Um kurze Briefe oder Notizen einzutippen, sind umfangreiche Textverarbeitungsprogramme wie Vizawrite, Mastertext oder Startexter viel zu aufwendig. Unser 20-Zeiler bietet die meisten Funktionen, die Sie benötigen. »Minitext« verträgt bis zu 20480 Zeichen in 256 Zeilen zu je 80 Zeichen: Der Bildschirm wird bei der Texteingabe horizontal und vertikal gescrollt! Außerdem wurde ein deutscher Zeichensatz mit entsprechender Tastaturbelegung eingebaut (Tabelle 1). Laden Sie das Programm mit:

```
LOAD "MINITEXT V1.0",8
```

Nach dem Start mit RUN dauert es ca. 50 s, bis das entsprechende Maschinenprogramm im Bereich ab $4000 (16384) generiert wird. Der Arbeitsbildschirm erscheint. Sie können mit der Eingabe Ihres Textes beginnen. Die Cursor-Tasten, <RETURN>, <DEL> und <INST> zeigen dieselbe Wirkung wie beim Basic-Editor. Mit <F1> ruft man in der obersten Bildschirmzeile ein Arbeitsmenü auf:

<S>ave: Text speichern (Filename angeben!),
<L>oad: Text-File laden,
<P>rint: Text auf Drucker ausgeben.

»Minitext« arbeitet mit allen ASCII-kompatiblen Druckern zusammen, die seriell mit dem C64 verbunden sind. Gerätenummer und Sekundäradresse sind voreingestellt (Zeile 10 im Listing: OPEN 4,4,0). Diese Werte können Sie in dieser Programmzeile jedoch beliebig ändern und an den eigenen Drucker anpassen. (Steffen Börm/bl)

Sonderzeichen bei »Minitext«

Taste	erzeugt Zeichen
<;>	ä
<SHIFT ;>	Ä
<:>	ö
<SHIFT :>	Ö
<@>	ü
<SHIFT @>	Ü
< < >:	Semikolon ;
< > >:	Doppelpunkt :
<Pfeil links>:	ß

Tabelle 1. Wie bei einer richtigen Textverarbeitung: »Minitext« kennt Sonderzeichen und Umlaute.

Der größte Basic-Speicher

38911 Byte freies RAM für Basic-Programme ist zwar eine ganze Menge, doch für umfangreiche Anwendungen manchmal zu knapp. »Extend Basic« schafft Platz: Das Utility verschiebt den Basic-Interpreter nach hinten und installiert diverse Hilfsroutinen. Diese kümmern sich um den Interrupt und leiten Sprünge des Betriebssystems um. Die neue Speicherbelegung zeigt Tabelle 2. Der Speicherbereich, den der Basic-Interpreter bisher belegte, schließt nun unmittelbar ans Basic-RAM an: Es besitzt nun stattliche 11 776 Byte mehr!

Laden Sie mit:

```
LOAD "EXTEND BASIC",8
```

und starten Sie es mit RUN. Es muß zunächst entpackt werden. Anschließend erscheint die Einschaltmeldung: 50687 Basic-Byte free. Daß dies kein Scherz ist, unterstreichen die Zeiger auf das Ende des verfügbaren Basic-RAM in den Adressen $37/ $38 (55/56):

```
PRINT PEEK(55) + 256 *
PEEK(56)
```

Als Ergebnis erhalten Sie »52736«. Allerdings muß man einige Nachteile in Kauf nehmen:
- kein Zugriff auf den RAM-Bereich von $CE00 bis $FFFF,
- die Basic-Vektoren und das NMI dürfen nicht verändert werden.

Möglich ist dagegen,
- die Speicherstelle 1 zu benutzen,
- den I-O-Bereich sowie den Zeichensatzgenerator ab $D000 anzusprechen,
- SYS-Befehle ins ROM auszuführen,
- den Bereich ab $C000 bis $CDFF zu sperren, um darin Routinen in Maschinensprache unterzubringen,
- alle Basic-Befehle wie bisher zu verwenden.

Mit »Extend Basic« stehen viele Anwendungsmöglichkeiten offen: Adventures verbessern, größere Datenstrukturen verwalten, Verminderung der »Garbage Collection« aufgrund des immensen Variablenspeichers und last not least, Programme zu entwerfen, die vorher nicht in den Speicher paßten. Das Programm läßt sich nach einem Reset mit »SYS 52960« reaktivieren. Wem die Entpackerei nach jedem Neustart des Programms zu lange dauert, sollte das Maschinensprache-File laden:

```
LOAD "EXTEND BASIC.MC",
8,1
```

Nach der Eingabe von »NEW« läßt sich die Erweiterung mit »SYS 49152« starten. (Olaf Kummer/bl)

Extend Basic (Speicheraufteilung)

Bereich			Funktion
$0801	bis	$CDFF	frei für Basic-Programme und Variablenspeicher
$CE00	bis	$CF7F	Hilfsroutinen
$CF80	bis	$CFFF	reserviert
$D000	bis	$DFFF	neuer Speicherbereich des Basic-Interpreters
$E000	bis	$E4FF	wie vorher, mit Hilfsroutinen
$E500	bis	$F4FF	ehemaliger Bereich von $B000 bis $BFFF
$F500	bis	$F5FF	reserviert
$F600	bis	$FF7F	frei
$FF80	bis	$FFFF	Sprungtabelle

▲ Tabelle 2. »Extend Basic« krempelt den Speicher des C64 völlig um



Drucker-Basic – eine komfortable Erweiterung

Druck ohne Druck

Tagelange Programmierarbeit scheint vergessen. Das Erfolgserlebnis ist da. Aber halt – da ist ja noch der Printer. Linearkanal hier, Grafik-Mode da. Wie bring ich's zu Papier? Kein Problem! Mit dem »Drucker-Basic«.

von Arno Baumfalk

Es gibt Basic-Erweiterungen für die verschiedensten Zwecke. Am bekanntesten ist wohl Simon-Basic oder auch unser Hypra-Basic, sogar ein eigenes Mailbox-Basic zum Konstruieren einer Mailbox existiert. Aber alle behandeln die Druckerausgabe mehr oder weniger stiefmütterlich. Doch was ein richtiger Drukker-Freak ist, dem ist das bei weitem zu wenig, denn viele

Befehl	Funktion	Parameter
OPN	Öffnen eines Drucker-Kanals zur Übertragung von Escape-Sequenzen	keine
CLS	Schließen des mit OPN geöffneten Kanals	keine
CR	Carriage-Return (Wagenrücklauf)	keine
PF a,n	Paper-Feed (Einstellen des Zeilenabstands)	Bei a=6 fällt n weg; 1/6 Zoll Zeilenabstand, bei a=8 fällt n weg; 1/8 Zoll, bei a=72 werden n/72 Zoll Zeilenabstand eingestellt, bei a=216 schließlich n/216 Zoll
LF	Line Feed (Zeilenvorschub)	keine
BACK n	Drucken und Papiertransport rückwärts	Rückwärtstransport um n/216 Zoll
AHEAD n	Drucken und Papiertransport vorwärts	Vorwärtstransport um n/216 Zoll
PAGE a,n	Seitenlänge in Zeilen oder Zoll festlegen	Bei a=1 wird die Seitenlänge auf n Zeilen festgelegt (1-127) Bei a=2 wird die Seitenlänge auf n Zoll festgelegt (1-22)
FF	Seitenvorschub	keine
JUMP a,n	Überspringen der Perforation	Bei a=0 fällt n weg; es wird kein Perforationssprung durchgeführt Bei a=1 wird ein automatischer Perforationssprung um n Zeilen durch geführt (1-127)
VT	Ausdrucken der im Puffer befindlichen Daten und Transport zu einer vorher festgelegten vertikalen Tabulatorposition	keine
VFU n	VFU Kanal bestimmen	Kanalnummer n (0-7)
RIGTH n	Rechten Rand setzen	n=Anzahl der Schreibstellen (1-255)
LEFT n	Linken Rand setzen	n=Anzahl der Schreibstellen (0-255)
HT	Ausführung vorher festgelegter horizontaler Tabs	keine
LARGE n	Breitschrift	n=0 bedeutet ausschalten; n=1 bedeutet einschalten
SI n	Komprimierte Schrift	siehe LARGE
EMPH n	Emphasized Printing (Fettdruck)	siehe LARGE
DOUBLE n	Doppeldruck	siehe LARGE
ELITE	Schriftart = Elite	keine
PICA	Schriftart = Pica	keine
SUPER n	Superscript	siehe LARGE
SUB n	Subscript	siehe LARGE
UNDERLINE	n Unterstreichen	siehe LARGE
MODE n	Bestimmung der Druckart (entspricht ESC!+n)	n=Druckart (0-61)
PROP n	Proportionalschrift	siehe LARGE

Befehl	Funktion	Parameter
INKR n	Inkrementales Drucken (Jedes Zeichen wird sofort gedruckt)	siehe LARGE
CHRERW n	Zeichenerweiterung (ESC I+n)	siehe LARGE
SELECT	Setzt Drucker in Select-Status	keine
DESELECT	Setzt Drucker in Deselect-Status	keine
DELLIN	Delete Line (Löschen einer Zeile)	keine
DELCHR	Delete Character (Löschen eines Zeichens)	keine
CS n	Frei ladbarer Zeichensatz	siehe LARGE
COPY	Kopieren des internen in den frei ladbaren Zeichensatz	keine
NLQ	Near Letter Quality	siehe LARGE
DENSITY a,n	Bit-Image-Graphik	Bei a=1 wird einfache Dichte eingestellt, bei a=2 doppelte Dichte, bei a=3 doppelte Dichte, doppelte Geschwindigkeit, bei a=4 vierfache Dichte n ist in jedem Falle die Anzahl der Daten
CRT a,n	Bit-Image-Graphik (CRT-Grafik)	Bei a=1 wird die CRT-Grafik mit 640 Punkten/8 Zoll eingestellt, bei a=2 mit 720 Punkte/8 Zoll. n=Anzahl der Daten.
PLOTTER n	Plotter — Grafik bestimmen (576 Punkte/8 Zoll)	n=Anzahl der Daten
NINE a,n	Neun-Nadel-Bit-Image-Grafik	Bei a=1 : einfache Dichte, bei a=2 : doppelte Dichte. n=Anzahl der Daten
INIT	Drucker initialisieren	keine
PAPER n	Papierendeerkennung	siehe LARGE
BEL	Summer	keine
BS	Backstep (Rückwärtsschritt)	keine
RET	Druckkopf in Ausgangsstellung	keine
UNI n	Unidirektionaler Druck	siehe LARGE
SLOW n	Halbe Druckgeschwindigkeit	siehe LARGE
ITALIC	Kursivschrift	siehe LARGE
USA	Internationalen Zeichensatz wählen	keine
FRANCE		
GERMANY		
BRITAIN		
DANMARK		
SWEDEN		
ITALY		
SPAIN		
JAPAN		
MSBOFF	MSB Steuerung löschen	keine
MSB n	Höchstwertiges Bit eines Datenwortes	Bei n=0 wird das Bit 0, bei n=1 wird es 1

Tabelle 1. Die neuen Befehle des »Drucker-Basic«

SONDERHEFT 66

VORSCHAU

15 Spiele der Extraklasse

erwarten Sie im 64'er-Sonderheft 66. Nicht nur der Daumen am Joystick, auch das logische Denkvermögen wird auf eine harte Bewährungsprobe gestellt:

- »Suburbia«, eine ferne Zukunftswelt im Jahre 2100, wird von einem interstellaren Erdbeben bedroht. Bewahren Sie die Bewohner vor der Katastrophe!
- Wären Sie gerne Astronaut? Dann ist »On the moon«, die spielfilmähnliche Simulation der Mondlandung 1969, genau das richtige für Sie! Sekundenbruchteile entscheiden darüber, ob der Riesenschritt für die Menschheit gelingt.
- Wetten, daß... es J.R. Ewing vor dem Einschlafen spielt? »Harte Dollars«, das Dallas-Monopoly für angehende Rinder-Barone und Ölbohrer, erfordert viel Grips und Kombinationsgabe.
- 25 neue Levels zu »Quadranoid« (aus dem letzten Spiele-Sonderheft), sowie Tips und Trainer-POKEs zu vielen bekannten C64-Games verwandeln die gute Stube in eine Spielhalle!



Nr. 66 gibt's ab 24.5.91 bei Ihrem Zeitschriftenhändler.

Aus aktuellen oder technischen Gründen können Themen verschoben werden. Wir bitten dafür um Verständnis.

Funktionen des Printers lassen sich leider nur sehr kompliziert und speicherplatzaufwendig programmieren. Hier hilft Drucker-Basic, denn es stellt 58 neue Befehle zur Verfügung, mit denen die Programmierung des Druckers ihre Schrecken verliert. Drucker-Basic ist hauptsächlich für die Besitzer von Druckern, die nach dem ESC/P-Standard angesprochen werden, programmiert. Andere Drucker akzeptieren leider nur einen eingeschränkten (je nach Drucker unterschiedlich) Teil der Befehle. Hier hilft nur Probieren. Die Besonderheit des Drucker-Basic liegt darin, daß es als Maschinenprogramm den normalen Ablauf eines Basic-Programms nicht verlangsamt und sich deshalb als Unterprogramm zur Druckersteuerung in eigene Programme einbinden läßt. Geladen wird es von der beiliegenden Diskette mit

```
LOAD"Drucker-Basic",8,1
```

und nach Eingabe von NEW mit SYS 40855 gestartet. Für Ihre Basic-Programme stehen Ihnen nun 38802 Byte und zusätzliche Kommandos zur Verfügung (Tabelle 1/S. 49).

Mit Hilfe des Programms »Interface« können die Kanalnummer, die Geräteadresse und die Sekundäradresse der Datei, die zur Übertragung der Anweisungen benutzt wird, verändert werden. Hier geben Sie die Werte ein, die Ihr Interface in den Direktmodus (es werden keine Codewandlungen durchgeführt) bringt. Voreingestellt sind hier:
Dateinummer: 255
Geräteadresse: 4
Sekundäradresse: 1

Um die Variablenzuweisung beim Drucker-Basic nicht unnötig zu verlangsamen, sollte man den LET-Befehl jetzt nicht mehr weglassen. Statt A=1 sollte bei Verwendung der Erweiterung LET A=1 geschrieben werden.

Assembler-Programmierern wird es sicher sehr gefallen, daß auch zusätzliche Kommandos in Drucker-Basic integriert werden können. In der Befehlstabelle wurde dafür noch Platz gelassen. Sie liegt, wie ein Großteil des Programms, ab $A001 unter dem Basic-ROM. Von $A702 bis $A200 steht noch Freiraum zur Verfügung. Die Wörter müssen in ASCII-Code eingegeben werden. Zum letzten Buchstaben wird $80 addiert. Die darauffolgenden 2 Byte stellen die Startadresse dar. Nun fehlt nur noch eine Änderung: das Low-Byte des Endes der Tabelle in $A282 und das High-Byte in $A289 müssen in die Adresse des letzten Buchstabens des letzten Befehlswortes umgeändert werden (es handelt sich um ein Zeichen, zu dem $80 addiert wurde). Die Fehlerbehandlung erfolgt mit JSR 9FD9 (Fehlernummer im Akkumulator). Die Get-Byte-Routine liegt bei 9FB8 (Ergebnis=X Reg.), die Get-Adreßroutine bei 9FCC (Ergebnis = $14/$15) und schließlich die Check-Kommaroutine bei 9FC2. Nach Ausführung eines Kommandos muß nach 9FA6 gesprungen werden. (gr)

Kurzinfo: Druckerbasic

Programmart: Drucker-Tool
Laden: LOAD"DRUCKER-BASIC",8
Starten: nach dem Laden RUN eingeben
Benötigte Blocks: 8
Programmautor: Arno Baumfalk



Das C 64 Tips & Tricks Poster

Umschlagseiten von den Heftklammern lösen!

POKE...

19, 64	INPUT ohne Fragezeichen
19, 0	INPUT mit Fragezeichen
650,128	Alle Tasten »Repeat«
650, 0	Alle Tasten »Normal«
768,185	Fehlermeldungen unterdrücken
768,139	Fehlermeldungen freigeben
808,225	<Run/Stop-Restore> sperren
808,237	<Run/Stop-Restore> freigeben
788, 52	<Stop> sperren
788, 49	<Stop> freigeben
775, 1	Listschutz einschalten
775,167	Listschutz abschalten
646, x	Zeichenfarbe bestimmen
53280, x	Rahmenfarbe bestimmen
53281, x	Hintergrundfarbe bestimmen

PRINT PEEK (...)

203	Code der aktuell gedrückten Taste
653	1: <Shift> 2: <Cbm> 3: <Ctrl>
649	Länge des Tastaturpuffers
198	Anzahl der gedrückten Tasten
678	1: PAL-Computer 2: NTSC-Computer

SYS...

64738	Kaltstart (wie Einschalten)
65126	Warmstart (wie <RUN/STOP-RESTORE>)
65511	alle offenen Files schließen

WAIT...

56321, 1, 1	Wartet auf »Oben« an Joystick-Port 1
56320, 1, 1	Wartet auf »Oben« an Joystick-Port 2
56321, 2, 2	Wartet auf »Unten« an Joystick-Port 1
56320, 2, 2	Wartet auf »Unten« an Joystick-Port 2
56321, 4, 4	Wartet auf »Links« an Joystick-Port 1
56320, 4, 4	Wartet auf »Links« an Joystick-Port 2
56321, 8, 8	Wartet auf »Rechts« an Joystick-Port 1
56320, 8, 8	Wartet auf »Rechts« an Joystick-Port 2
56321, 16, 16	Wartet auf »Feuer« an Joystick-Port 1
56320, 16, 16	Wartet auf »Feuer« an Joystick-Port 2
653, 1	Wartet auf <Shift>
653, 2	Wartet auf <Cbm>
653, 4	Wartet auf <Ctrl>
198, 1	Wartet auf Drücken einer Taste

▲ **PEEKS, POKES und SYS-Anweisungen, die jeder Basic-Programmierer stets zur Hand haben sollte.**

Alle 29 Register des Soundchips (SID) für die Tonerzeugung. Rechts sind die jeweiligen Funktionen beschrieben. ▼

BIT 7	BIT 0	Registername	
F1	F0	Frequenz low	Stimme 1
F9	F8	Frequenz high	
P1	P0	Pulsweite low	
P9	P8	Pulsweite high	
nchronisat.	GATE	Kontrollregister	
ecay 1	Decay 0	Attack/Decay	
elease 1	Release 0	Sustain/Release	
F1	F0	Frequenz low	Stimme 2
F9	F8	Frequenz high	
P1	P0	Pulsweite low	
P9	P8	Pulsweite high	
nchronisat.	GATE	Kontrollregister	
ecay 1	Decay 0	Attack/Decay	
elease 1	Release 0	Sustain/Release	
F1	F0	Frequenz low	Stimme 3
F9	F8	Frequenz high	
P1	P0	Pulsweite low	
P9	P8	Pulsweite high	
nchronisat	GATE	Kontrollregister	
ecay 1	Decay 0	Attack /Decay	
elease 1	Release 0	Sustain /Release	
GF 1	GF 0	Grenzfrequenz low	Filter
GF 4	GF 3	Grenzfrequenz high	
lter 2	Filter 1	Resonanz/Filter	
L 1	L 0	Mode/Lautstärke	

t X 1	Pot X 0	Potentiometer X
t Y 1	Pot Y 0	Potentiometer Y
01	00	Oszillator 3
H1	H0	Hüllkurve Osz. 3

Adresse hex	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$ D000	X-Position des Sprite Nr. 0							
$ D001	Y-Position des Sprite Nr. 0							
$ D002	X-Position des Sprite Nr. 1							
$ D003	Y-Position des Sprite Nr. 1							
$ D004	X-Position des Sprite Nr. 2							
$ D005	Y-Position des Sprite Nr. 2							
$ D006	X-Position des Sprite Nr. 3							
$ D007	Y-Position des Sprite Nr. 3							
$ D008	X-Position des Sprite Nr. 4							
$ D009	Y-Position des Sprite Nr. 4							
$ D00A	X-Position des Sprite Nr. 5							
$ D00B	Y-Position des Sprite Nr. 5							
$ D00C	X-Position des Sprite Nr. 6							
$ D00D	Y-Position des Sprite Nr. 6							
$ D00E	X-Position des Sprite Nr. 7							
$ D00F	Y-Position des Sprite Nr. 7							
$ D010	Spr. 7, msb X-Pos.	Spr. 6, msb X-Pos.	Spr. 5, msb X-Pos.	Spr. 4, msb X-Pos.	Spr. 3, msb X-Pos.	Spr. 2, msb X-Pos.	Spr. 1, msb X-Pos.	Spr. 0, msb X-Pos.
$ D011	msb des Rasterregisters (Reg. 18)	Schaltbit für veränderten Hintergrundfarbmodus 1 = eingeschaltet	Schaltbit für Hochauflösungsmodus 1 = eingeschaltet	Schaltbit für Bildschirm »aus« 0 = normaler Bildschirm 1 = Bildschirmfarbe	Schaltbit für Zeilenzahl 0 = 24 Zeilen 1 = 25 Zeilen	Wert der Zeilenverschiebung in Y-Richtung beim Smooth Scrolling		
$ D012	Rasterregister. Dazu kommt das msb in Bit 7, Register 17							
$ D013	Lichtgriffel X-Position							
$ D011	Lichtgriffel Y-Position							
$ D015	Ein- und Ausschalten von Sprites. 0 = Sprite aus, 1 = Sprite an							
	Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
$ D016	(unbenutzt)		Reset-Bit, muß 0 sein, damit VIC-II-Chip arbeitet	Schaltbit für Mehrfarbmodus 1 = eingeschaltet	Schaltbit für Spaltenzahl 0=38 Spalten 1=40 Spalten	Wert der Spaltenverschiebung in X-Richtung beim Smooth Scrolling		
$ D017	Sprite-Vergrößerung in Y-Richtung. 0 = normale Größe, 1 = doppelte Größe							
	Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
$ D018	Startadresse Textbildschirm				Startadresse Zeichengenerator oder Hires-Bitmap			(unbenutzt)
$ D019	Interrupt-Flaggen-Register Interrupt				Lichtgriffel-Interrupt-Flagge	Sprite/Sprite-Kollision	Sprite/Hintergrund-Kollision	Raster-Interrupt-Flagge
$ D01A	Interrupt-Masken-Register Interrupt				Lichtgriffel-Interrupt-Maske	Sprite/Sprite-Koll.-Maske	Sprite/-Hintergrund-Kollision-Maske	Raster-Interrupt-Maske
$ D01B	Sprite/Hintergrund-Prioritätenregister. 0 = Sprite hat Priorität, 1 = Hintergrund hat Priorität							
	Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
$ D01C	Sprite-Mehrfarbmodus-Register. 0 = Normaldarstellung, 1 = Mehrfarbmodus-Darstellung							
	Sprite 3.5	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
$ D01D	Sprite-Vergrößerung in X-Richtung. 0 = normale Größe, 1 = doppelte Größe							
	Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
$ D01E	Sprite/Sprite-Kollision. 0 = keine Berührung, 1 = Berührung							
	Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
$ D01F	Sprite/Hintergrund-Kollision. 0 = keine Berührung, 1 = Berührung							
	Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
$ D020	(unbenutzt)				Farbe des Bildschirmrahmens			
$ D021	(unbenutzt)				Hintergrundfarbe Nr.0 (normale Hintergrundfarbe)			
$ D022	(unbenutzt)				Hintergrundfarbe Nr. 1			
$ D023	(unbenutzt)				Hintergrundfarbe Nr. 2			
$ D024	(unbenutzt)				Hintergrundfarbe Nr. 3			
$ D025	(unbenutzt)				Sprite-Mehrfarben-Register Nr. 0			
$ D026	(unbenutzt)				Sprite-Mehrfarben-Register Nr. 1			
$ D027	(unbenutzt)				Sprite 0, Farbe			
$ D028	(unbenutzt)				Sprite 1, Farbe			
$ D029	(unbenutzt)				Sprite 2, Farbe			
$ D02A	(unbenutzt)				Sprite 3, Farbe			
$ D02B	(unbenutzt)				Sprite 4, Farbe			
$ D02C	(unbenutzt)				Sprite 5, Farbe			
$ D02D	(unbenutzt)				Sprite 6, Farbe			
$ D02E	(unbenutzt)				Sprite 7, Farbe			

▲ **Die Register des VIC-II-Chip (msb = höchstwertiges Bit einer Adresse)**

DAS ERSTE REINRASSIGE VIDEO-SPIELE-MAGAZIN!



ab 27. März 1991 im Zeitschriftenhandel!